Guideline

Transformer in Production

Product(s): PowerPlay Transformer

Area of Interest: Modeling

Copyright

This document is maintained by the Best Practices, Product and Technology team. You can send comments, suggestions, and additions to cscogpp@ca.ibm.com .

Contents

Purpose

Demands for larger and more complex PowerCubes are becoming a common occurrence as businesses grow and expand. As this occurs, optimizing build times and runtime performance becomes extremely important.

The purpose of this document is to provide guidance and 'best practice' methodology to aid in performance related strategies concerning PowerCube build and runtime performance.

This document spans several versions of Transformer up to and including Series 7 Version 2. We advise you to confirm specific version capabilities before embarking on a project.

Audience

This document is intended for an advanced audience that should have a thorough understanding in all or most of these areas:

- PowerPlay Transformer
- RDBMS structures and servers
- Dimensional modeling
- UNIX or NT performance tuning
- UNIX or NT hardware

The information in this document has been gathered during numerous investigations concerning real life client models and experiences. It is important to understand that not all guidelines or considerations will always have the same effect due to the endless variations of model types and client environments.

Overview

With the introduction of cube optimization changes in Transformer 6.0 it became possible for users to build larger, better performing cubes in a shorter time frame. Most of the time these new changes will provide a benefit without the need for any interaction on the part of the PowerCube designer. In other cases, the size and complexity of the models and cubes can require intervention on the part of the cube designer/builder in order to streamline the cube building process and create the best performing cubes possible. Without experience or guidance the process of streamlining a cube build can be a tedious process. Quite often choices made before any model or cube is even built can have an impact on the final build time and performance of the PowerCubes.

Exceptions

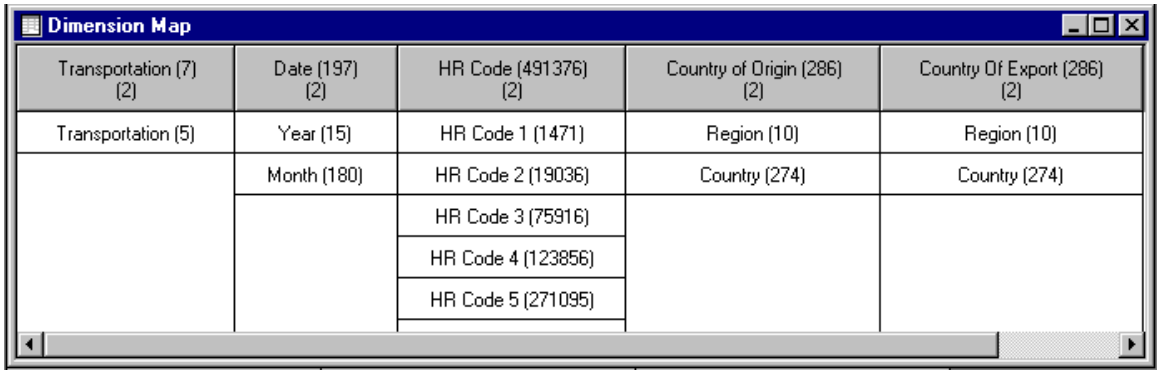
This document was written using Transformer 6.0. Although the performance tuning guidelines in this document are still applicable to newer versions of the product, the actual settings may vary depending on newer and more powerful hardware configurations.

As of IBM Cognos 8.x, the information contained in this document can be found in the core documentation.

Test Model Information

Throughout the following pages you will find the results of various tests that were performed on different computers and platforms. A brief description of the test model used *"Category and Row Dominant Model"* is included here for reference purposes.

Dimension Map and Model Attributes:



Model Attribute	Description
Number of Categories	492,152
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 (two calculated), 3 x 64 bit, 2 after-rollup calculated
Source Data Format	ASCII ('~' delimited)
Number of source files	9 (6 are structural, 3 are transactional) Enable Multi-Processing was set for the 4 largest data sources.
Number of Transaction input records	50 million
Size (in MB) of all source files	2,280 (2.28GB)

This model was used to create a PowerCube with the following settings:

- Auto-partitioning with 5 passes maximum and a maximum partition size of 500,000 records
- Crosstab caching enabled
- All tests were run with the default Transformer settings unless specified otherwise

Data Related Considerations

Analyzing the source data is a valuable step to determine the quality, storage, source type and the preparation that is required.

Clean and Consolidate Your Data

Preprocessing your data will bring additional performance benefits:

- Read time will be faster in Transformer if the source data only contains the information required for the model. For example, if additional columns that are not used are included in a data source, Transformer will take additional time to process the columns, even though they are not used in the model.

- Data consolidation reduces the number of records to be read. The lower the number of records Transformer has to read, the shorter the PowerCube build time.

Designing OLAP Models

Nothing will improve build or runtime performance in a PowerPlay application to the same extent as taking the time to design your OLAP model and PowerCubes well. How you define your models can affect both the size of the model and the processing time required to generate PowerCubes.

This section describes issues to consider during the design stage.

Structural and Transactional Data Sources

Restructuring your source data into separate structural and transactional data sources will reduce processing loads in Transformer.

Structural data sources contain columns whose values build the dimensional hierarchies within a Transformer model. The data provided by these data sources are associated with the dimensions and levels of the model and provide the data that is used to generate categories, category structures, labels, sort values, descriptions, and so on.

For the best performance possible, we recommend that each dimension or drill-down path be defined with a separate structural data source. In the Data Source dialog box in Transformer, the structural data sources should be defined in order that they are used to build the dimensions and levels from left to right. This allows Transformer to process one data source, update a dimension with the categories and then continue on to the next data source and dimension. If the data sources are not in the correct order and Cube optimization is not set to Default (auto-partition), Transformer may have to reprocess a data source that it has previously processed in order to build a dimension.

Transactional data sources provide the measure values needed for the PowerCube. Columns in a transactional data source are associated with measures and with unique levels in the model. The last data sources listed should be the transactional data sources to provide measures for the dimensions.

Transactional data sources change frequently, usually by representing the latest data that is to be added to the PowerCube. These data sources should be designed to have small, concise records with the absolute minimum amount of information required to add new data to the PowerCubes.

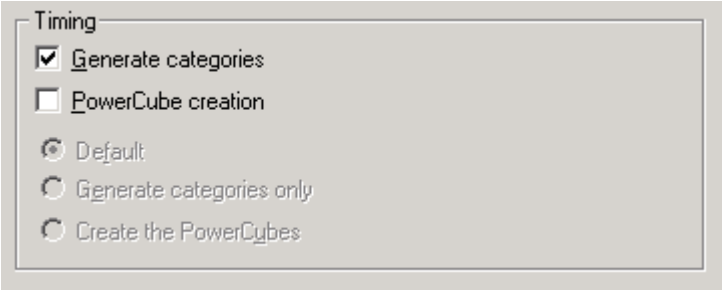
Tips

- When designing data sources that will be used in Transformer, minimize the amount of processing by only including the columns required to build the model. If you include additional columns that are not required, you may impact the data source processing time.
- When possible, maintain category structures within Transformer models to eliminate the redundant processing required to continually rebuild them.
- If long descriptions are included in the model, we recommend that you generate PowerCubes using models that are already populated with categories associated with the descriptions.

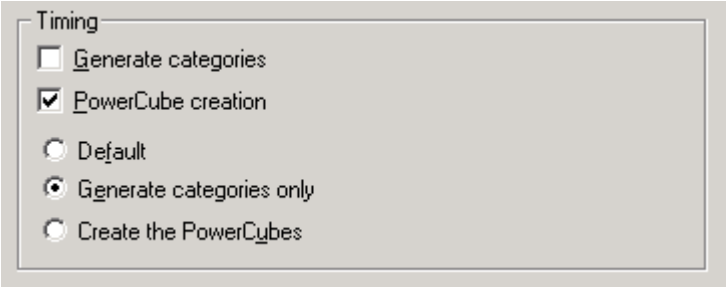
Timing

Timing controls (Data Source property sheet) can be set to control when Transformer processes each data source.

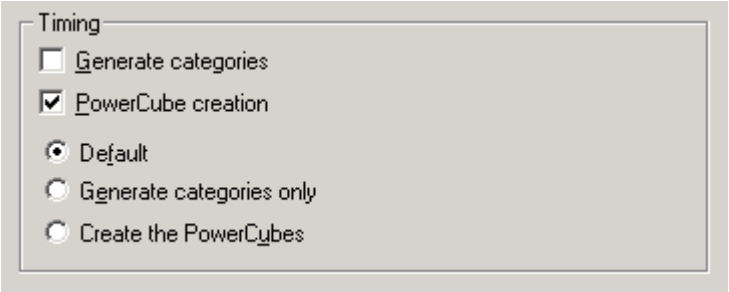
Structural data sources should be executed initially to build the category structure within the model. Once this is done, if there is no requirement to execute them during the PowerCube generation (no new categories have been added to the data source and the model has been saved populated with these categories) then the Timing for that data source can be set as follows:



Some structural data sources represent a more volatile structure that requires the categories to be updated each time a PowerCube is generated. The timing for data sources of this type can be set to run during the category generation phase of PowerCube creation:

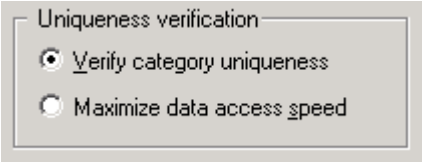


Transactional data sources are constantly changing with new data required for the measure values each time a PowerCube is generated. Transactional data sources are executed during PowerCube creation, to provide the measure values:



Verify Category Uniqueness vs. Maximize Data Access Speed

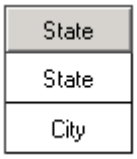
In the data source property sheet there are two settings for Uniqueness Verification. By default this attribute is set to Verify Category Uniqueness. This setting is recommended for data sources that provide columns that are associated with levels in a dimension containing unique levels. Typically these are structural data sources.

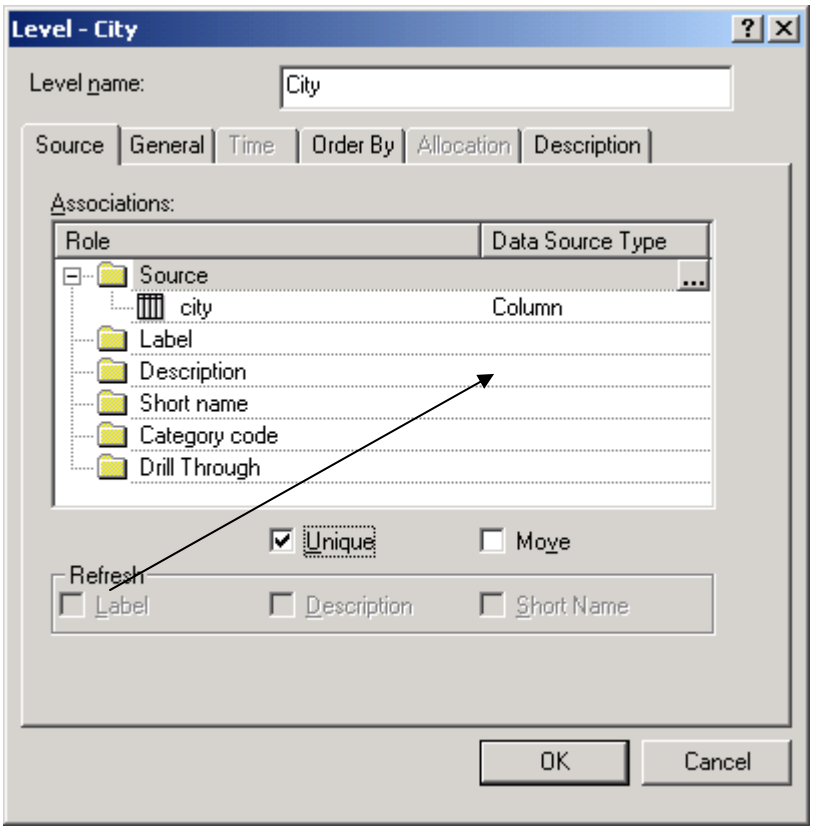


If Verify Category Uniqueness is set and Transformer detects two categories with the same source value on a level that has been identified as Unique (Level properties), the following errors will be returned:

> (TR2317) The level 'City' is designated as unique. Source value 'Green Bay' was used in an attempt to create a category in the path (By state,Illinois,Green Bay). 'Green Bay' already exists in level 'City' in the path (By state,Wisconsin,Green Bay).
>
> (TR0136) A uniqueness violation was detected. The process has been aborted.

For example, the State dimension has Unique set on the City level:

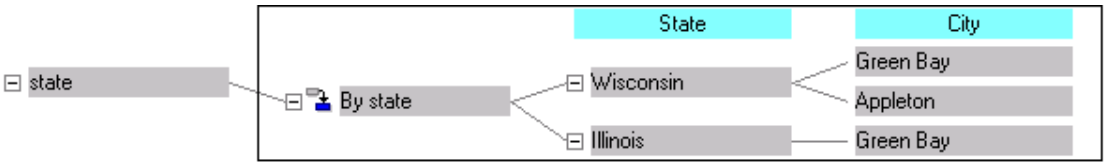




The error indicates that the second instance of Green Bay under the City level exists (in this case Illinois). For example, if you have the following as source data:

Measure, State, City
1, Wisconsin, Green Bay
2, Wisconsin, Appleton
3, Illinois, Green Bay

When Unique is not checked on the City level you will see the following in the dimension diagram:

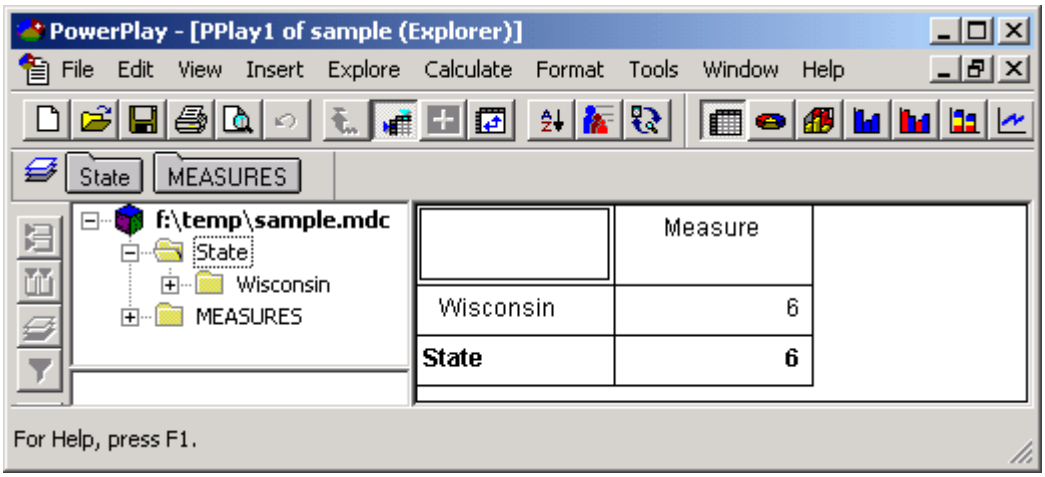


When Unique is checked on the City level the process is aborted and the dimension diagram displays:

If you are certain that the values in the model data sources do map into unique categories in a level, the Maximize Data Access Speed attribute can be set. When invoked, uniqueness validations are minimized and performance of data source processing improves. Transformer will not have to continuously verify category values against existing values, which could mean a significant performance improvement. This is especially true for transactional data sources with columns for many levels in which all of the level columns are included in the data source.

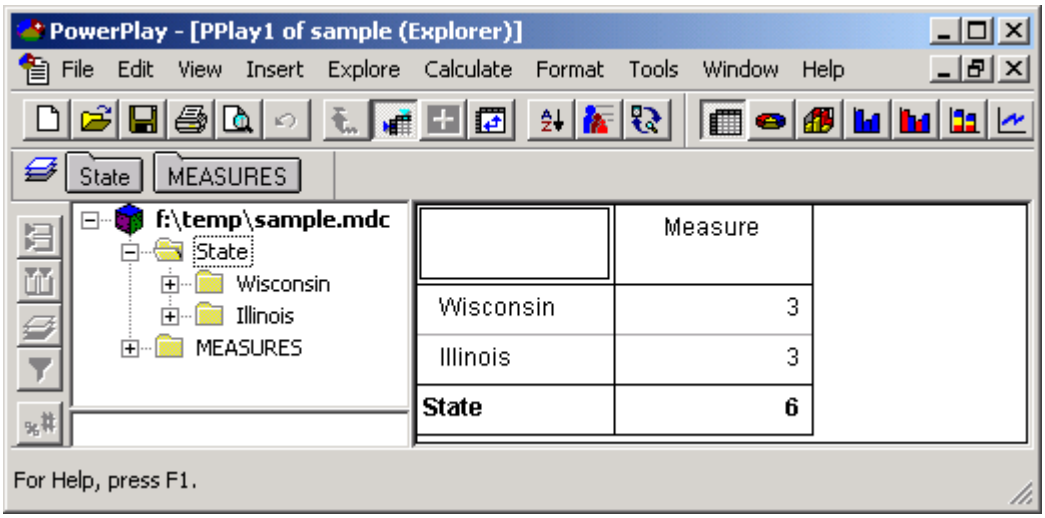
Warning! If Maximize Data Access Speed is enabled and a uniqueness violation exists in your data, Transformer will not notify you. The end result will be missing categories and incorrect values in the PowerCube.

Using the same example above, if Maximize Data Access Speed is enabled with Unique set on the City level, Transformer will not notify you that Green Bay exists under two different States (Wisconsin and Illinois) and the end result in PowerPlay will be the following:



Notice that Illinois doesn't exist in the crosstab above.

If Unique is removed from the City level and the cube is rebuilt, the end result in PowerPlay will be the following:



NOTE: Unique moves will not be performed when Maximize Data Access Speed is set.

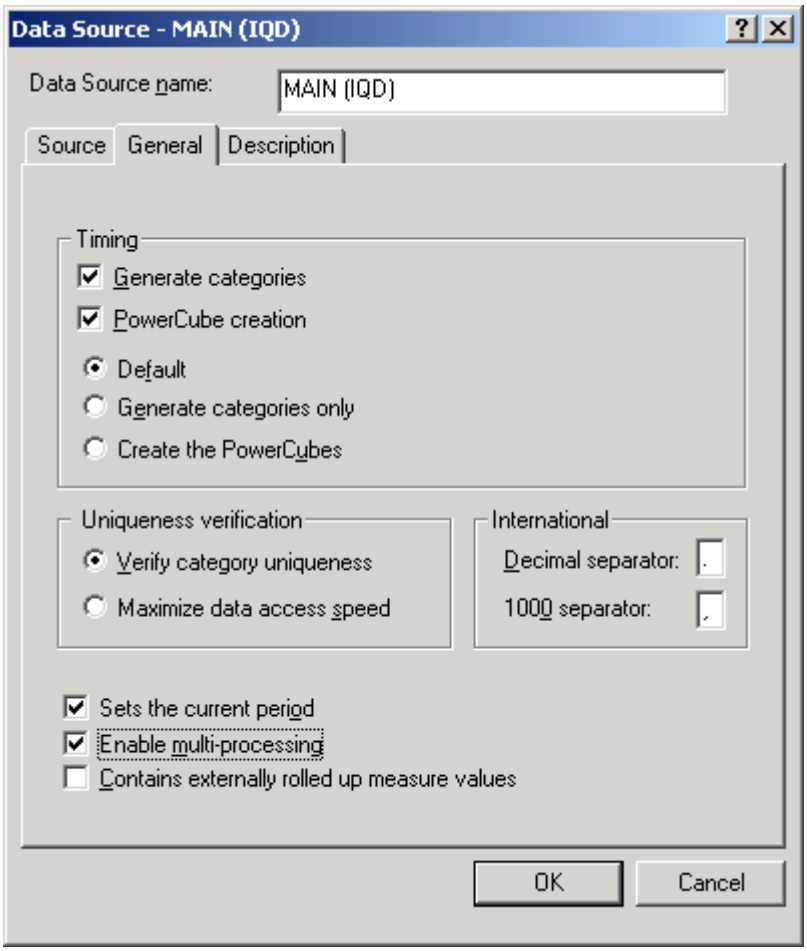
Multi-Processing

If dual CPUs are available on the computer that builds the PowerCubes, you can take advantage of the Multi-Processing feature. Enabling this feature can significantly improve the overall performance of your PowerCube builds during the Data Read phase.

Multi-Processing is only enabled for the following data source types:

- Impromptu Query Definition (IQD)
- Delimited Field Text
- Delimited Field Text with Column Titles
- RDBMS Sources via an Architect Package

This option is set on a query-by-query basis in the Data Source property dialog box:



Partitioning

The goal of partitioning has always been to achieve a satisfactory end user query response time without exceeding the PowerCube production window. When considering the production window, decreasing partition size generally improves query response time by reducing the volume of data required for query calculation. However, this improvement is at the expense of increasing the cube build time.

General Guidelines

Both Manual and Auto-partitioning utilize a new cube build algorithm that was introduced in version 6.0. However there are some features available within Transformer that *disable* Auto-partitioning. The features that prohibit the use of the new cube build algorithm are:

- PowerCube Optimization is not set to Auto-partition
- Consolidation is set to: either of No or Yes with Presort
- Using Externally rolled up measures
- Using Before Rollup calculated measures
- Cloaking a Primary Drill category

A warning message in the log file or during a Check Model will alert you to the fact that auto-partitioning is not being utilized. An example of this error message is:

> (TR2757) This model contains one or more cubes that use a dimension view in which the primary drilldown is cloaked. Auto-partitioning is not possible when a primary drilldown is cloaked.

Warning! Disabling auto-partitioning could result in a severe degradation of performance during the cube build phase. One extreme case of performance degradation was a cube build that took almost 8 hours to complete. Once auto-partitioning was used, the same cube built in less than an hour.

As a rule of thumb, if a PowerCube contains more than 250,000 records, partitioning should be defined to speed up runtime access for end users. Partitioning will pre-summarize the data in the PowerCube and group it into several subordinate partitions so retrieval will be significantly faster. Creating a very large cube without the use of partitioning can result in poor runtime performance for the end users. However as the number of partitions increase, the longer it will take to create the cube.

Performance gains with partitioning are most noticeable when models have hierarchical dimensions and levels, which are maintained with a parent to child ratio of about 1:10. Partitioning will not perform well in models that contain hundreds of categories at the top level of a dimension with several thousand categories at the level directly beneath it.

It is important to note that a partition level is not the same as a level in a dimension. A partition level is a set of categories that receive the same partition number. These categories can come from more than one level in a dimension.

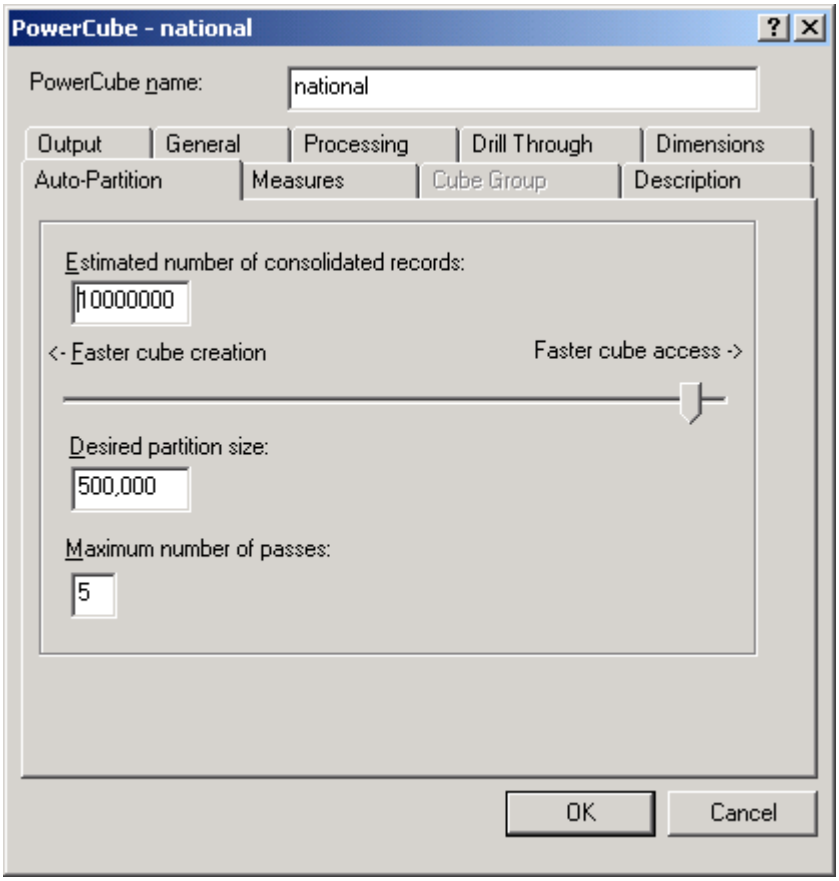
Most models have dimensions where the lowest level details are frequently accessed, such as product codes. In dimensions such as these, it is important to manage categories with a high parent to child ratio and partition them accordingly. Normally auto-partitioning will do a good job provided you pick a partition size that is large enough. The term ‘large enough’ can be defined as the number of records required to satisfy most end user queries. Too many partition levels will adversely affect lowest level detail reports.

Auto-Partitioning

When auto-partition is set as the Optimization method for a PowerCube, Transformer will pre-summarize records to a summary partition while leaving lower level detail records in partitions that you specify. In doing this, the summarization required at run-time is dramatically reduced which means the response time for end-users is significantly improved.

Auto-Partitioning Settings

The following displays the PowerCube auto-partition tab and a list of settings that can be adjusted:



- Estimated Number of Consolidated Records: This setting specifies the estimated number of records a cube will contain. Transformer uses consolidation to combine records that contain identical non-measure values, which will result in a smaller cube with improved runtime performance. To find out how many consolidated records are in a PowerCube, build the PowerCube once with default settings and check the log file for "End Count and Consolidation with..” as indicated in the sample log below:

  --- Performing Pass 4 with 11465545 rows and 8542 categories remaining.
  Selected Dimension 1 for next pass of partitioning.
  Counting category hits.
  End sorting 11465545 records.
  Start Count and Consolidation with 11465545 rows and 8708 categories remaining.
  End Count and Consolidation with 7312012 rows and 8542 categories remaining.

  You can then set the Estimated Number of Consolidated Records in the PowerCube auto-partition dialog box.

- Faster Cube Creation/Access: Specifies the desired partition size. Setting this towards Faster Cube Creation will decrease performance for the end user but will shorten the cube creation time. Setting this towards Faster Cube Access will

increase the cube build time but will also improve performance for the end user.

- Desired Partition Size: This setting is based on the Estimated Number of Consolidated records setting. Transformer uses this setting to select categories that satisfy the partition size you specify and that optimize query performance.

- Maximum Number of Passes: Transformer will use up to the number of passes you specify to determine the best query performance when partitioning. One pass is done for each partition level that is created. As you decrease the desired partition size and increase the number of passes, the number of partition levels created increase, which increases the cube creation time.

Transformer Log File

The sample Transformer log file below will show you where auto-partitioning is performed in the model:

--- **Performing Pass 0** with 22770681 rows and 8708 categories remaining.
Selected **Dimension 3** for next pass of partitioning.
Sorting the work file.
Counting category hits.
End sorting 22770681 records.
Start Count and Consolidation with 22770681 rows and 8708 categories remaining.
End Count and Consolidation with 22770681 rows and 8708 categories remaining.
Start Write leaving 8708 categories remaining.
Updating the PowerCube data.
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 8000000.
Performing DataBase Commit at record number 10000000.
Performing DataBase Commit at record number 12000000.
Performing DataBase Commit at record number 14000000.
Performing DataBase Commit at record number 16000000.
Performing DataBase Commit at record number 18000000.
Performing DataBase Commit at record number 20000000.
Performing DataBase Commit at record number 22000000.
Performing DataBase Commit at record number 22770682.
End Write leaving 8708 categories remaining..
Timing
--- **Performing Pass 1** with 22770681 rows and 8708 categories remaining.
Selected **Dimension 11** for next pass of partitioning.
Counting category hits.
End sorting 22770681 records.
Start Count and Consolidation with 22770681 rows and 8708 categories remaining.
End Count and Consolidation with 15522151 rows and 8708 categories remaining.
Start Write leaving 8708 categories remaining.
Updating the PowerCube data.
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 8000000.
Performing DataBase Commit at record number 10000000.
Performing DataBase Commit at record number 12000000.
Performing DataBase Commit at record number 14000000.
Performing DataBase Commit at record number 15522152.
End Write leaving 8708 categories remaining..
Timing
--- **Performing Pass 2** with 15522151 rows and 8708 categories remaining.
Selected **Dimension 0** for next pass of partitioning.
Counting category hits.
End sorting 15522151 records.
Start Count and Consolidation with 15522151 rows and 8708 categories remaining.
End Count and Consolidation with 14848450 rows and 8708 categories remaining.
Start Write leaving 8708 categories remaining.
Updating the PowerCube data.

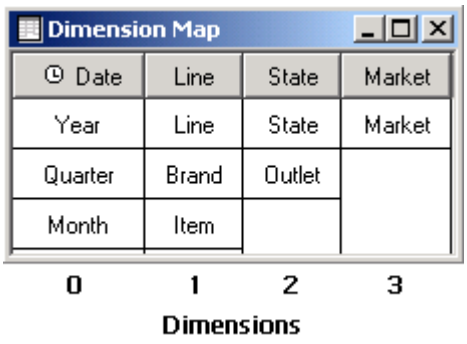
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 8000000.
Performing DataBase Commit at record number 10000000.
Performing DataBase Commit at record number 12000000.
Performing DataBase Commit at record number 14000000.
Performing DataBase Commit at record number 14848451.
End Write leaving 8708 categories remaining..
Timing
--- **Performing Pass 3** with 14848450 rows and 8708 categories remaining.
Selected **Dimension 0** for next pass of partitioning.
Counting category hits.
End sorting 14848450 records.
Start Count and Consolidation with 14848450 rows and 8708 categories remaining.
End Count and Consolidation with 11465545 rows and 8708 categories remaining.
Start Write leaving 8708 categories remaining.
Updating the PowerCube data.
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 8000000.
Performing DataBase Commit at record number 10000000.
Performing DataBase Commit at record number 11399138.
End Write leaving 8708 categories remaining..
Timing
--- **Performing Pass 4** with 11465545 rows and 8542 categories remaining.
Selected **Dimension 1** for next pass of partitioning.
Counting category hits.
End sorting 11465545 records.
Start Count and Consolidation with 11465545 rows and 8708 categories remaining.
End Count and Consolidation with 7312012 rows and 8708 categories remaining.
Start Write leaving 8708 categories remaining.
Updating the PowerCube data.
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 7312013.
End Write leaving 8542 categories remaining..
Timing
--- **Performing Pass 5** with 7312012 rows and 8542 categories remaining.
Start Write leaving 8542 categories remaining.
Updating the PowerCube data.
Updating the PowerCube data.
Performing DataBase Commit at record number 2000000.
Performing DataBase Commit at record number 4000000.
Performing DataBase Commit at record number 6000000.
Performing DataBase Commit at record number 7312013.
End Write leaving 8542 categories remaining..
Timing

Analyzing a Transformer Log File

The sample Transformer log file (above) shows a total of 5 passes where Transformer attempts partitioning (passes 0-4) with the final pass (pass 5) where Transformer performs summary partition consolidation.

When auto-partitioning is used, Transformer determines which dimensions to partition based on an internal algorithm. In the sample Transformer log file, dimensions 3, 11, 0 and 1 are used. Transformer made two passes on dimension 0 which provided additional consolidation on the second pass.

Dimension 0 is actually the first dimension in the dimension map in Transformer so when you compare the dimensions listed in the log file to the dimensions in the Transformer model, you always need to keep in mind that the log file counts the first dimension as 0 as shown below:



To determine if a model is being partitioned well, compare Pass 0 to Pass 5 (or the last Pass listed in the log file):

--- **Performing Pass 0** with 22770681 rows and 8708 categories remaining.

--- **Performing Pass 5** with 7312012 rows and 8542 categories remaining.

Comparing the Passes above you can see that auto-partitioning is working well for this model as the original row count of 22770681 was consolidated down to 7312012. You will also notice that a hierarchical consolidation was done where the original category count 8708 was consolidated down to 8542. The end result will mean fewer partitions will need to be created.

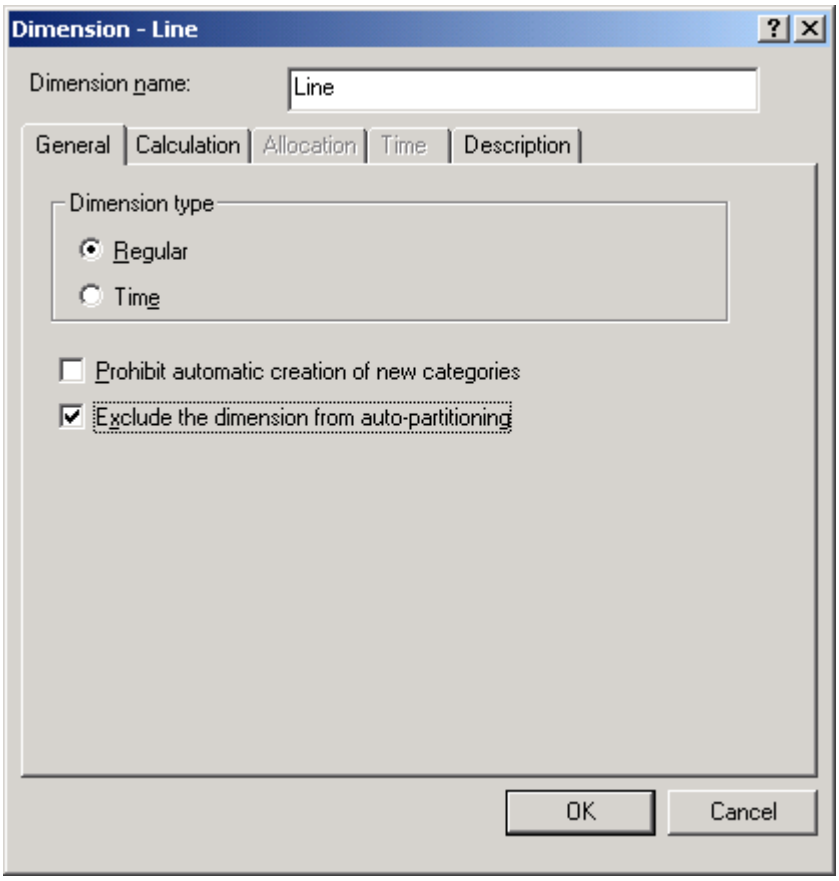
Excluding Dimensions

It is possible to exclude dimensions from auto-partitioning. Dimensions might be considered for exclusion for the following reasons:

- Dimension is large and flat
- Dimension is frequently reported on at the lowest level
- Dimension has alternate drilldowns

By excluding large detail dimensions it will be possible to set the partition size smaller, which means cube build time and summary queries will be faster. If the same model included partitioning on detail dimensions, lowest level queries would be slower since they are reporting across multiple partitions.

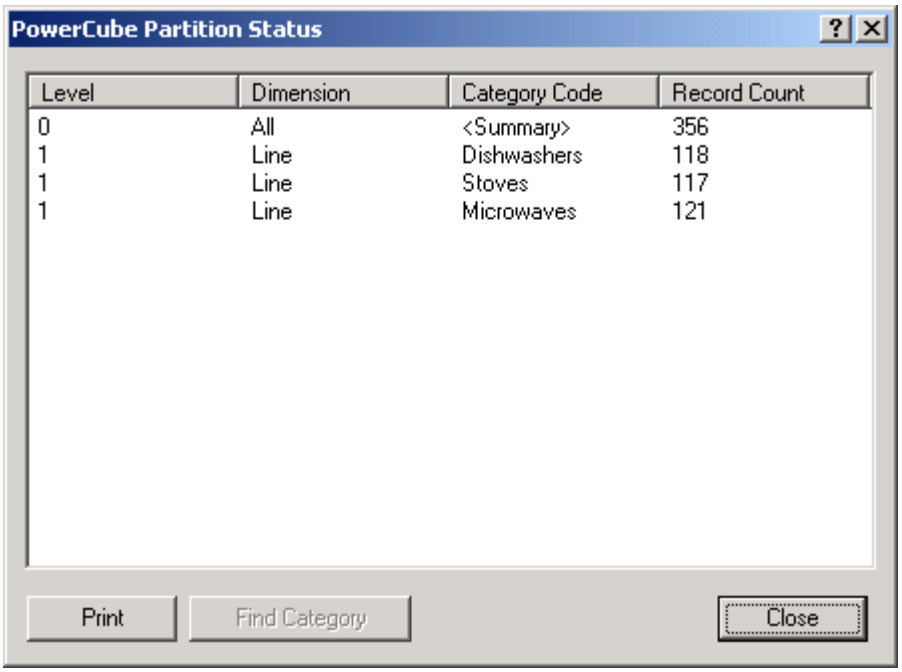
To exclude a dimension, open the dimension property sheet and enable Exclude the Dimension from auto-partitioning:



Tips

- Analyzing the log file will tell you whether the number of records and categories are decreasing from pass to pass. If the number of records doesn't decrease on the last pass, try reducing the Maximum Number of Passes by one to eliminate the last pass or increase the desired partition size.
- To arrive at an effective partitioning strategy try starting with the default partition size of 500,000 and then increase it to a large number (such as 1,000,000 or 1,500,000). Depending on the results, you may try increasing the number again or if the results are unsatisfactory, you can decrease the number in increments until you achieve the best partitioning strategy.
- Transformer will use up to the number of passes you specify and will always stop partitioning when the number of records in the summary partition drop below the specified partition size.
- If both the summary partition and the level one partition have the same number of records then the summary partition has not been consolidated. Try increasing the Maximum Number of Passes.

- To see the current partitioning information of a PowerCube, right click the cube in the PowerCube list in Transformer and select PowerCube Partition Status:



The PowerCube Partition Status window will display the partition levels, dimensions, category codes and record counts where partitioning has been applied.

Manual Partitioning

The auto-partition optimization introduced in Transformer 6.0 was designed to dynamically look at the cube structure and data and then determine a suitable partition strategy. In the majority of cases this will result in good query performance for PowerCube consumers and there is no need to apply partitioning manually. In large and complex models however, the PowerCube might not perform as expected and it then becomes important to be able to determine what the current partitioning strategy is so that the cube can be optimized. This section will provide some guidelines for manually partitioning PowerCubes.

Manual partitioning is generally used for the following reasons:

- Attempt to improve on a current auto-partitioning strategy
- Your cubes are large or unusually structured
- You want to achieve performance tuning for specific reports

It is sometimes possible to improve on auto-partitioning strategies. If your models have large flat dimensions, you may prefer to use manual partitioning to specify where partitioning is done on the dimensions, levels and categories. An example of a flat dimension is illustrated below:

Once you have assigned a partition number to any level, auto-partitioning is disabled. Only the levels and categories specified manually will then be considered for partitioning. Transformer will take the levels and categories you choose for partitioning and determine if they are suitable candidates for partitioning.

You must still define the Maximum Number of Passes property on the auto-partition tab of the PowerCube property sheet. The number of passes defined must be set to the number of partition levels assigned or greater.

A partition level is a set of categories that all receive the same partition number. These categories can come from more than one level in a dimension but we recommend that you select them from the same level and include a specific set of categories across the dimension.

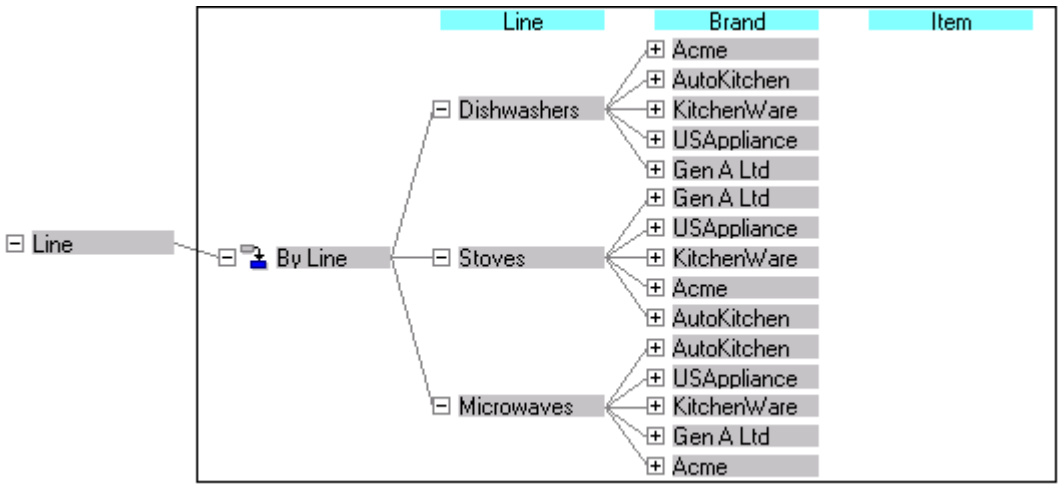
If you have a dimension where the lowest level categories are accessed frequently, it is especially important to maintain levels with a parent-to-child ratio of 1:10 or less.

You must decide whether to favor summary reports or lowest level detail reports when designing a partitioning strategy, as only one partition size can be specified for any given cube. If your end users generally create summary reports with the categories from a specific dimension, consider partitioning at a high level in that dimension.

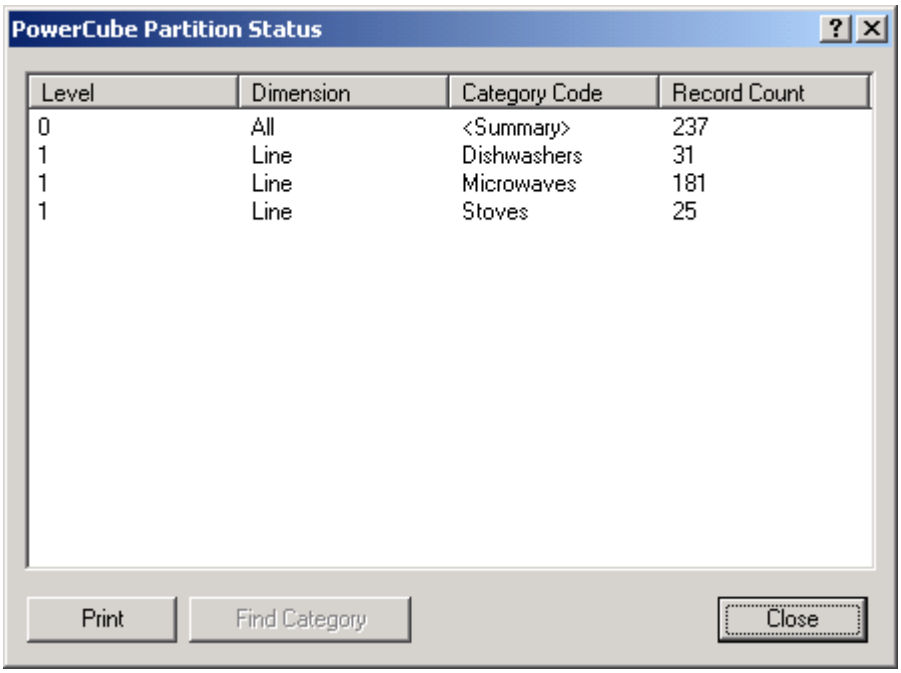
Category Partitioning

After defining level partitioning you may notice in the PowerCube Partition Status window that a large number of records is contained under one or more parent categories. Adding individual categories from the child level below to the same partition may optimize partitioning.

Using the diagram below we will be focusing on the Dishwasher, Stove and Microwave Lines and their child categories under Brand:



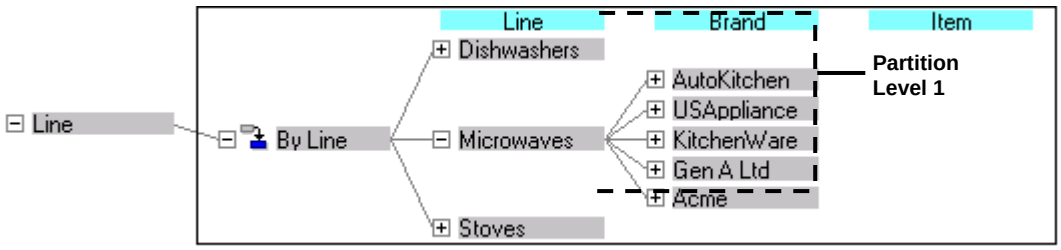
After applying a single partition on the Line level, the PowerCube Partition Status window displays the following record counts:



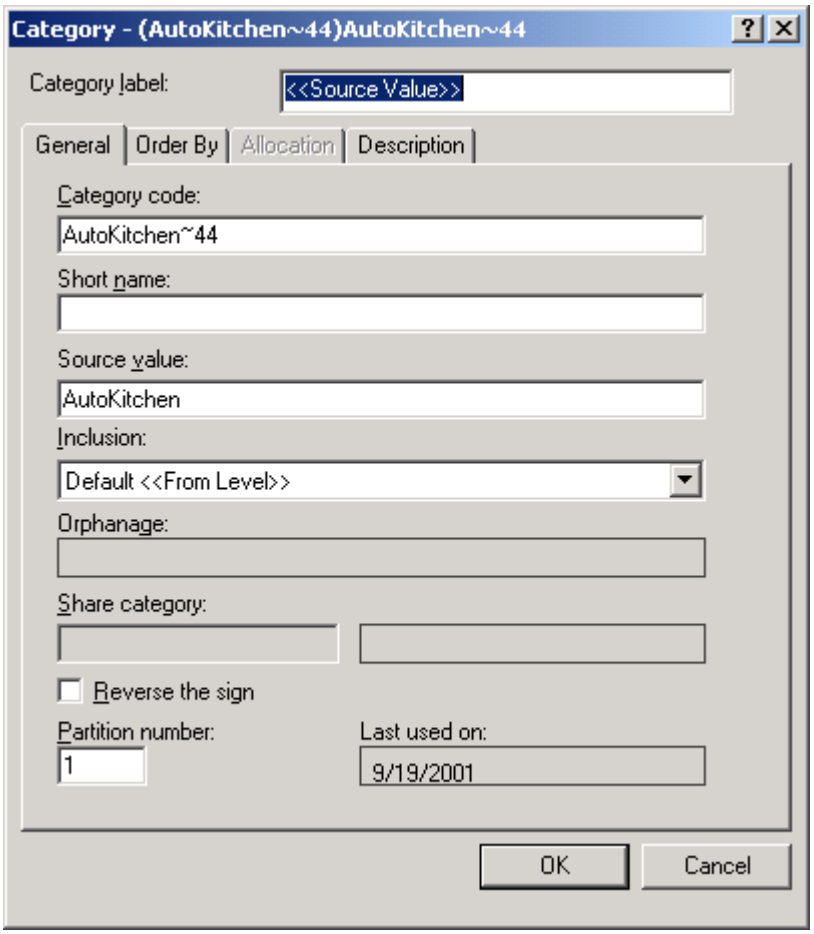
The PowerCube Partition Status window (above) displays the Line category codes listed for Dishwashers, Stoves and Microwaves. Notice that Microwaves contains a record count of 181, Dishwashers contains 31 and Stoves contains 25. In relative terms (meaning if the record counts were larger, such as 181,000, 25,000 and 31,000), this would mean that a query might perform well when you drill down on Dishwashers and Stoves. However, when you drill down on Microwaves the query response time might be much slower because of the large number of records contained in this partition.

To evenly distribute the total number of categories under the Line dimension, you can individually assign the same partition number to all of the Microwave child categories

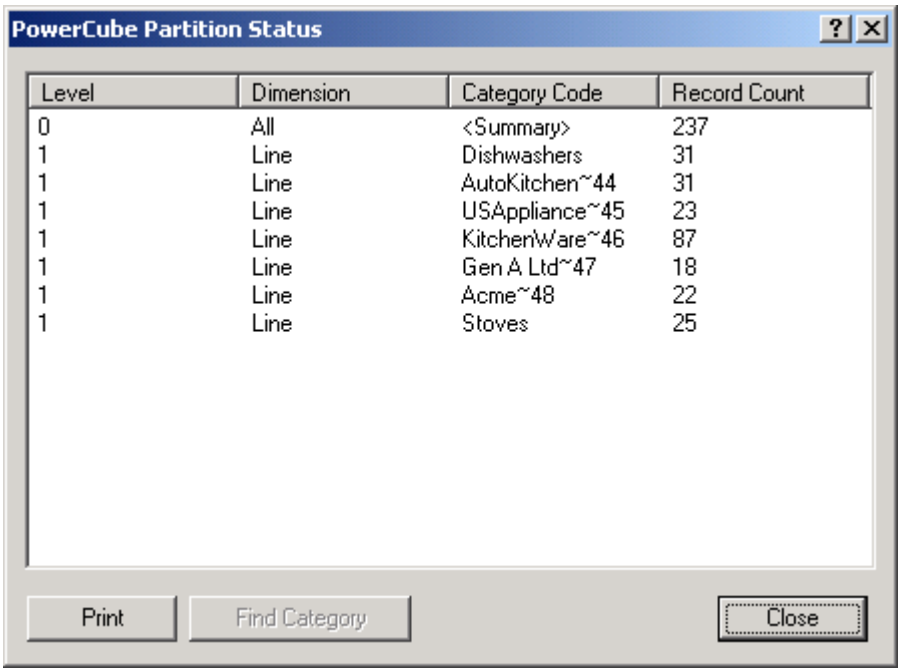
in the Brand level. For example, in the diagram below you can see that Dishwashers and Stoves in the Line level and the Microwave Brand child categories in the Microwave Line have been added to partition level one:



The Properties dialog box for each category in the Brand level would have Partition 1 assigned to it as can be seen in the Partition number property in the following dialog box:



After applying these changes and rebuilding the cube, the resulting PowerCube Partition Status window displays that partition level one encompasses the Dishwasher and Stove Line levels as well as Microwaves child categories in the Brand level:



Determining a Partitioning Strategy

Use the following steps to help you to determine a partitioning strategy for your model:

1. Select the dimension that contains the largest number of categories. In addition, consider dimensions that contain many levels in comparison with other dimensions in the model. Such dimensions most often offer the greatest potential for row consolidation during cube processing.

2. Choose a desired partition size, expressed as the number of records. This size is chosen to optimize runtime performance against the cube. Typically, partitions should not contain more than 250,000 records.

3. Use the number of rows in your data source to calculate the number of partitions you will require to obtain the desired partition size. This becomes the set of partitions in a partition level. The following calculation can be used:

   number of partitions = number of source rows/desired partition size

   Note: A maximum of 3 partition levels in a model usually derives the best results.

4. In the selected dimension, choose a level that contains approximately as many categories as the number of partitions determined in Step 3.

5. In the first partition level, assign partition number 1 to each category in the chosen level. To assign the same partition number to each category of a level, assign a partition number to that level. In other partition levels, assign a partition

number larger than the number assigned to categories in the previous partition level.

Note: A partition level is not the same as a level in a dimension. A partition level is a set of categories that receive the same partition number. These categories can come from more than one level in a dimension. For simplicity, it is best to choose an entire level to which you can assign the same partition number. In practice, when calculating the number of partitions in Step 3, you would try to select a set of categories across the entire dimension and assign these categories the same partition number. Once complete, you should not be able to traverse a path from the root category to a leaf level without encountering a category with the partition number assigned.

6. Build the cube, then right-click the cube to review the partition status. If the size of any partition is too large, another level of partitioning may be necessary. If the partition status is unacceptable, that is, some partitions contain more records than the desired partition size, proceed to test the performance in PowerPlay in Step 7. If the partition status is acceptable, no further steps are required.

7. Navigate the cube in PowerPlay, drilling down into the partition with the largest number of records. If the performance is unacceptable, consider another level of partitioning in these dimensions. In Transformer, examine the number of records in the summary partition. This is the number of records that you have to consider in subsequent partitioning.

8. Go to Step 3 and repeat the entire partitioning process using a level (in some other dimension) that adequately divides the number of records in the summary partition. For each new level of partitioning, increase the partition number by 1. As a rule, avoid using more than two levels of partitioning to generate cubes. If you do not obtain the desired performance characteristics with two levels, consider using a different dimension or level to define your partitions.

9. After changing the partitioning, recreate the PowerCube and re-examine the partition status.

10. Repeat Steps 3 through 7 until there are a sufficient number of partition levels to yield desired runtime performance.

Tips

- Ensure that the majority of queries that the end users will want are answered from the first or upper partitions (called summary partitions).

- Avoid partitioning on a dimension that is constantly updated with new categories. For example, partitioning a time dimension is not effective, as each build can add new date categories to the model. It is best to partition on ‘static’ dimensions whose categories do not change from build to build.

- A PowerCube will have to be completely rebuilt if you change the partitioning scheme in a category or level.

- Avoid partitioning on dimensions containing alternate drilldowns.

- More than three levels of partitioning should not be required. If the results are unacceptable consider using another dimension or level.

- Arrange partitions so that the information required for most queries can be found within a single partition. Avoid partitioning a dimension for which lowest level detail reports are needed. Access times can be slower for these

reports than for summary level report because data is returned from several partitions.

- Leaf or special categories can't be used as partition categories.

When Assistance is Required with Partitioning

If your PowerCube is not providing adequate query performance then it is important that the following items/information be available in order to conduct an investigation:

- The Transformer log file from the PowerCube build. The log file contains information on which dimensions were involved in partitioning as well as details about how well the partitions are consolidating.
- The populated Transformer model (.MDL or .PYI) and PowerCube (.MDC). Together these files will provide a partition status that details all of the categories involved in partitioning along with record distribution.
- It is important to have a good understanding of which dimensions in the PowerCube are the most involved in queries. How deep the queries tend to be in the dimension is also important as this can suggest certain partition settings. In other words it is important to know whether consumers tend to query high up in the PowerCube or lower down, using which dimensions and levels.
- Accurate information on the queries being performed that are not meeting expectations. This should take the form of common reports or explorer activity with details about dimension line settings and/or nesting activity. If the model contains alternate drill hierarchies this should be clearly specified.

Dimension Views vs. User Class Views

Deciding whether to use Dimension Views or User Class Views can have significant impact on the size of your PowerCubes so it is important to understand the differences between these two features.

Dimension Views allow you to summarize, cloak, and apex categories in the dimensions included in PowerCubes. In addition, both dimension views and dimension diagrams allow you to exclude and suppress categories. These features are designed to reduce the number of categories that are to be placed into a cube, which in turn reduces the size of the PowerCubes. However this usually means that several small PowerCubes may have to be built compared to one large PowerCube.

User class views are used in conjunction with Access Manager. User class views do not remove categories from a PowerCube. Instead, they restrict access to categories for members of specific User Classes. This allows multiple users to have access to the same cube while allowing them to only see the data they are entitled to see. User class views are designed for large cubes shared among numerous users.

PowerCube Optimization

This feature specifies the method Transformer uses to optimize the PowerCube creation process. The auto-partition method usually provides the best solution for Transformer to devise a partitioning scheme. This is the default optimization setting.

For models that were created prior to 6.0, the auto-partition feature was not available. If existing models are not using the auto-partition method, we recommend that you

consider changing the PowerCube optimization method to auto-partition to explore the possible benefits that may be gained.

For further information on auto-partitioning, please refer to section 5.5.2.

Consolidation

Consolidation will reduce the size of the PowerCube by combining records with identical non-measure values into a single record, which will reduce the PowerCube size and shorten access time in PowerPlay.

There are four consolidation settings to specify whether Transformer should consolidate data in the PowerCube and the recommended setting to use is the Default setting.

Warning! Auto-partitioning usually doesn't perform well with Consolidation settings other than the Default setting.

Incremental Update

If the PowerCube production build window is not large enough to build the cube entirely, Incremental Update may be the answer. Incremental Update only adds the newest data to an existing PowerCube without reprocessing the previous data. Subsequently the PowerCube updates are much smaller than the entire rebuilding of the PowerCube and can be done much quicker.

You should only consider the use of the Incremental Update feature if the structure (dimensions, levels, etc) of the PowerCube remains static. If a structural change occurs, the cube must be regenerated from scratch with all of the data.

It is also recommended that you periodically rebuild the PowerCube from scratch. The first time a cube is built, auto-partitioning divides the dimensions and levels into various partitioning levels. Subsequent runs will add all new categories into partition level "0". If many categories are added over time, eventually the PowerCube consumer may notice performance problems. Rebuilding the PowerCube from scratch with all of the current categories will allow Transformer to devise a new partitioning scheme. The following is an example of a full rebuild schedule after every four incremental updates:

Build	Activity
1	Initial Load
2	Increment 1 on build 1
3	Increment 2 on build 2
4	Increment 3 on build 3
5	Increment 4 on build 4
6	Full Load consisting of the initial Load and Increments 1 through 4
7	Increment 5 on build 6
8	Increment 6 on build 7...

MultiFileCubes

It is becoming more and more common for companies dealing with enormous amounts of data to be forced to provide the end users with larger dimensional views of their data. In doing this it is possible to exceed the 2GB limit imposed on single file cubes. In Transformer version 6.6 and above, a cube can be split into multiple files when a cube contains more than a specified number of data records.

Transformer determines the number of output files needed, taking the number of data records in the cube, dividing by the threshold, and rounding up. The model must be partitioned as the partitions are spread evenly across multidimensional partition (.MDP) files, and an additional multidimensional cube (.MDC) file is added to hold the PowerCube metadata.

By default the MultiFileCube Threshold setting in the [PowerPlay Transformer] section of the trnsfrmr.ini file is set to zero which disables multifile cube generation. This threshold setting can be changed so large cubes are automatically output to multiple files. If your cube is less than 2GB try setting the threshold to 30,000,000:

```
[PowerPlay Transformer]
MultiFileCubeThreshold=30000000
```

If you want to use the MultiFileCube feature on smaller cubes, try setting it to a smaller number:

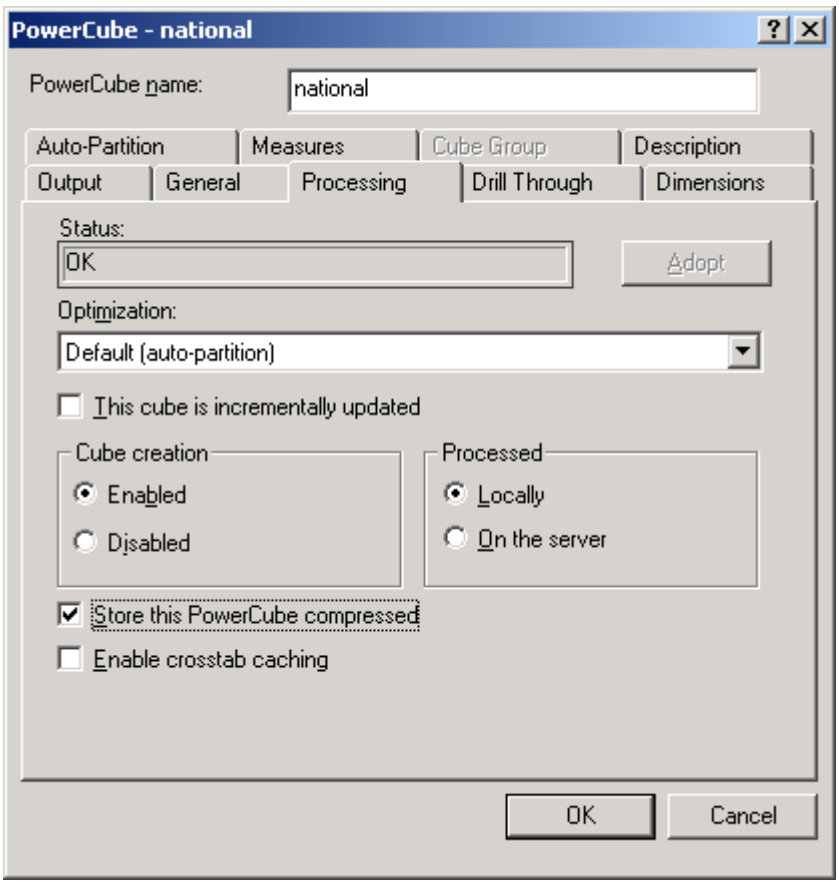
```
[PowerPlay Transformer]
MultiFileCubeThreshold=1000000
```

If you have a total of 90,000,000 records in the cube and you set the MultiFileCubeThreshold to 30,000,000, three .MDP and one .MDC file will be created.

Note: It is still possible to exceed the 2GB file size limit after setting MultiFileCubeThreshold property. If problems occur, decrease the setting in increments (try 20,000,000 or 10,000,000) until all resulting .MDP and .MDC files are less than 2GB in size.

Compressed PowerCubes

You can dramatically reduce the size of a PowerCube by selecting the Store this PowerCube Compressed property. This is beneficial if you are transferring files across the network. This option can be found in the PowerCube property sheet under the Processing tab:

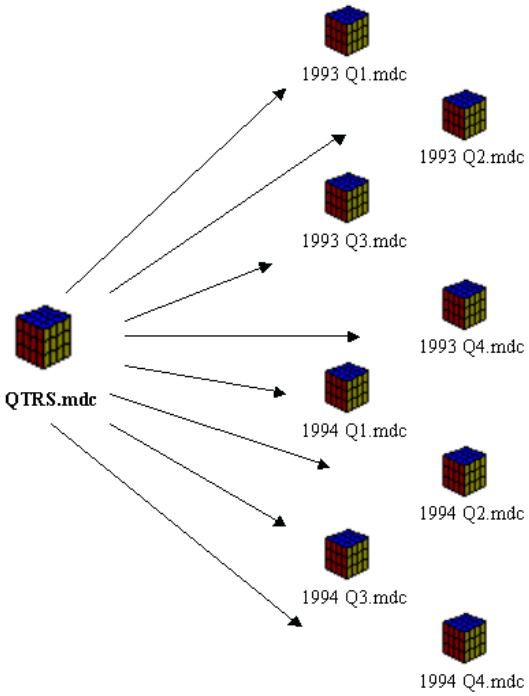


Compressed cubes have the same file extension as a regular PowerCube (.MDC). The first time a compressed cube is accessed via a PowerPlay application, the cube automatically decompresses without any user intervention. However this will result in the first query taking longer to return, as the cube has to be uncompressed first. To avoid this, use a macro to uncompress the cubes after they are copied over as part of a batch maintenance program.

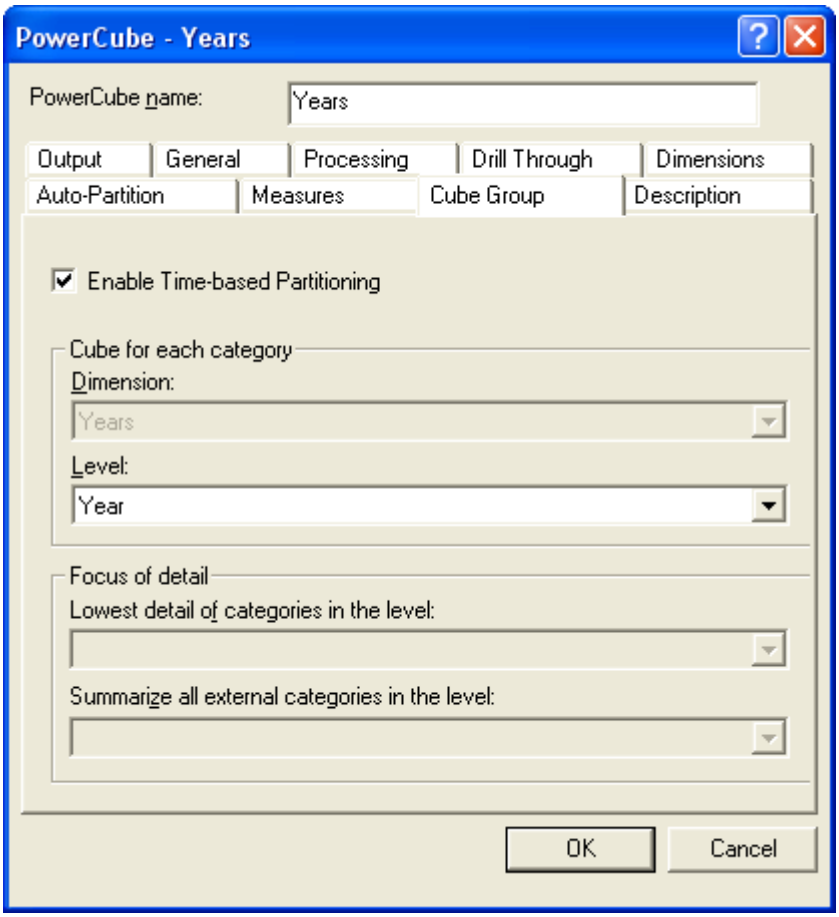
Time-Based Partitioned Cubes

Time-Based Partitioned Cubes are a new deployment option available in IBM Cognos Series 7 Version 2 for time segmented data updates and can be used as an alternative approach to using the Incremental Update feature.

Similar to a cube group, Time-Based Partitioned Cubes are made up of a collection of child cubes. The differing factor is that Time-Based Partitioned Cubes allow you to access the collection of child cubes simultaneously via a control cube. The child cubes can also be accessed independent of the control cube.



To implement the Time-Based Partition functionality, a cube group must be defined based on the time dimension and the Time-Based Partitioned Cube flag must be enabled.



These settings will define the child cubes based on the time dimension (at the appropriate level of the time dimension) and then combine them with a control cube so that the report consumers may access large amounts of related data via a single cube.

In comparison to incrementally updated cubes, the Time-Based Partitioned Cube may be easier to deploy and provide faster updating and reporting performance. For example, new data may be added every month based on the sales for that month. At the end of each month, a new "month" cube would be created and linked to the Time-Based Partitioned Cube. Report consumers are then able to report across the whole time dimension or across one month only to meet their requirements.

A Time-Based Partitioned Cube consists of the following elements:

- Multiple child cubes related to specific time periods at the same level of time granularity (e.g. month)
- A control cube which physically links the child cubes together and acts as the single point of connection for PowerPlay users
- A Time-Based Partitioned Cube definition file (.VCD), which references each child cube and their physical location. The definition file is an ASCII file, which can be manually edited by an administrator if required.

Advantages with Time-Based Partitioned Cubes

- Time-Based Partitioned Cube updates tend to be much faster in comparison to incrementally updated cubes as the new data is added in the form of a smaller single well-partitioned child PowerCube, rather than adding data to a single large standard PowerCube.
- Slowly changing dimensions are possible as the existing child cubes contain the history while newly created cubes can be created taking advantage of the category Move feature.
- A reduction in service interruption on a live system may also be possible in the physical time it takes to copy a smaller child cube compared to a large single cube.
- Rolling time support can be achieved by manually editing the definition file to remove pointers for the cubes that are no longer required. This will result in the categories being dropped for the report consumer. For new categories and cubes, Transformer will automatically update the control cube and definition file to append the latest references.
- By default, Time-Based Partitioned Cubes only relate to one specific level of time granularity (month or quarter for example). However it is possible for an administrator to reference various levels of time in the same model and cube. For example, one year cube, three quarter cubes and one month cube. This can be done for performance benefits.
- As the report consumers drill lower into the time dimension, performance will improve as they are accessing fewer cubes. As they reach the level of time granularity that the cubes are split on (month or quarter for example), performance will improve further as they are now only accessing one cube.
- Time-Based Partitioned Cube builds will result in a decrease in build time in comparison to incremental cubes. Incrementally updated cubes need to be rebuilt from scratch periodically whereas Time-Based Partitioned Cubes do not. This results in a more manageable update schedule.

Restrictions

- Time-Based Partitioned Cubes cannot be defined if a calculated category exists at a dimension level.
- New categories can be added to the existing child cubes; however, special categories or links to special categories are not created.
- Category count measures are not permitted.
- A Time-Based Partitioned Cube cannot be defined if more than one time dimension exists in the model.
- The Focus of Details options are disabled when the Time-Based Partitioned Cube option is selected.
- External rollup measures are not permitted.
- All measures must scope down to the Time-Based Partitioned Cube level defined or lower.

Slowly Changing Dimensions

Prior to Time Based Partitioned Cubes it was impossible to add data to an existing cube and allow existing categories to be moved within a dimension. Time-Based Partitioned Cubes allow categories to move within a dimension while maintaining the category code identification. This is referred to as a slowly changing dimension.

For example, based on the crosstab below, Marthe Whiteduck has sales for 2001 and 2002 in Vancouver.

		2001	2002	Years
Montreal	Henri LeDuc	$55,409	$33,021	$88,430
	Montreal	**$55,409**	**$33,021**	**$88,430**
Toronto	Lisa Testorok	$39,788	$24,156	$63,944
	Toronto	**$39,788**	**$24,156**	**$63,944**
Vancouver	Marthe Whiteduck	$11,700	$40,701	$52,401
	Vancouver	**$11,700**	**$40,701**	**$52,401**
Canada		**$106,897**	**$97,878**	**$204,775**

Marthe then moves from Vancouver to Ottawa. In the crosstab below you will see that Marthe now has the historical data for 2001 and 2002 in Vancouver and also has new data for 2003 in Ottawa.

		2001	2002	2003	Years
Montreal	Henri LeDuc	$55,409	$33,021	na	$88,430
	Montreal	**$55,409**	**$33,021**	**na**	**$88,430**
Toronto	Lisa Testorok	$39,788	$19,802	na	$59,590
	Toronto	**$39,788**	**$19,802**	**na**	**$59,590**
Vancouver	Marthe Whiteduck	$11,700	$40,701	na	$52,401
	Vancouver	**$11,700**	**$40,701**	**na**	**$52,401**
Ottawa	Marthe Whiteduck	na	na	$15,070	$15,070
	Ottawa	**na**	**na**	**$15,070**	**$15,070**
Canada		**$106,897**	**$93,524**	**$15,070**	**$215,491**

Adding and Removing Child Cubes

To add a cube, simply supply Transformer with the new data records. A new child cube will be generated and the corresponding control cube and definition files will be updated.

There are a few options available to remove an existing cube:

- To remove a cube, edit the definition file, remove the unnecessary cube references and delete the existing child cube.
- Exclude the category in the model so no child cube is created.

Altering Historical Data

Historical data can be manipulated by adding or subtracting data from existing cubes. For example, on January 14th a contract is signed for a total price of $12,451.00. In February it is decided that a rebate of $1,237.00 should be given. In order to reflect this rebate in the existing January cube, a negative amount of –1,237.00 is included in the data source for January 14th. The end result is a final sale price of $11,214 for January 14th in the cube.

Multi Level Time-Based Partitioned Cube

It is possible to create a Time-Based Partitioned Cube that is based on several levels of the time dimension; for example, one year cube, three quarter cubes and one month cube. This can be achieved by defining 3 Time Based Partitioned Cubes in the same model, generating the cubes and then editing the definition file to contain only the appropriate cube references. For example, the definition file would look like:

```
cube "2001" .\test\2001.mdc
cube "2002 Q1" .\test\2002 Q1.mdc
cube "2002 Q2" .\test\2002 Q2.mdc
cube "2002 Q3" .\test\2002 Q3.mdc
cube "October" .\test\October.mdc
```

Five cubes would now be accessed via the control cube instead of the original eight cubes as defined above. This could result in improved performance for the report consumer.

Editing the Definition Files

The definition file (.VCD) is an ASCII file that resides in the same directory as the control cube and also maintains the same filename as the cube.

For example, if the name of the Time-Based Partitioned Cube defined is “quarters.mdc”, the definition file will be created as “quarters.vcd” and a subdirectory containing all of the child cubes will be created as “quarters”.

To edit this file simply open it in any text editor:

```
cube "2001 Q1" .\quarters\2001 Q1.mdc
cube "2001 Q2" .\quarters\2001 Q2.mdc
cube "2001 Q3" .\quarters\2001 Q3.mdc
cube "2001 Q4" .\quarters\2001 Q4.mdc
cube "2002 Q1" .\quarters\2002 Q1.mdc
cube "2002 Q2" .\quarters\2002 Q2.mdc
cube "2002 Q3" .\quarters\2002 Q3.mdc
cube "2002 Q4" .\quarters\2002 Q4.mdc
```

The path defined in the definition file is relative to the placement of the control cube as indicated by the leading dot (.) below:

```
cube "2001 Q1" .\quarters\2001 Q1.mdc
```

A full path to the child cubes can be specified if required:

cube "2001 Q1" c:\Transformer\quarters\2001 Q1.mdc

Hardware and Environment

Hardware and environment settings can have a huge impact on performance during the cube build process and can also be the root cause of production related issues. This section is essentially a 'best practice' guide that focuses on selecting and enhancing a Transformer build computer through the use of hardware and environment settings.

Processor Considerations

Choosing the fastest available processor speed should be considered. The addition of a second CPU can result in a significant reduction in the data read phase when using Transformer's multi-processing feature.

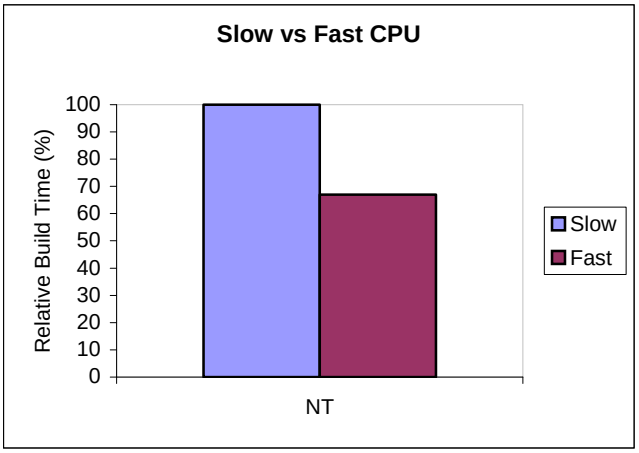
The data source type utilized in a model will impact the total reduction time when adding a second CPU. Using ASCII data sources will provide the greatest reduction in read time followed by reading RDBMS sources.

It is important to note that even though the fastest CPU should be selected, Transformer is not primarily a CPU bound application. If a bottleneck occurs during a PowerCube build it usually involves either the system memory or hard drive.

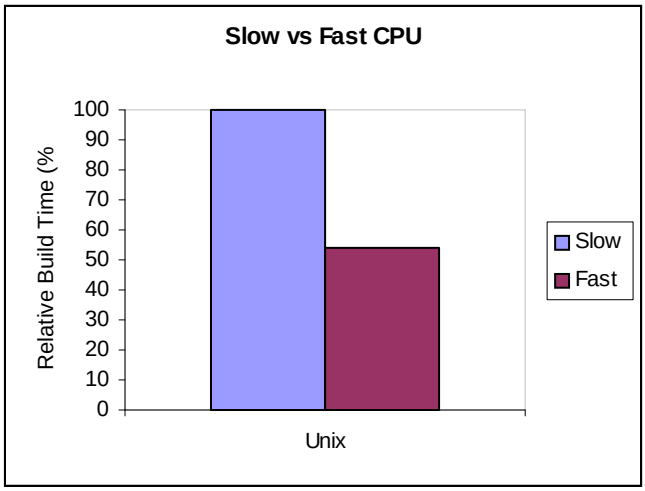
Slow vs. Fast CPU Build Examples

Using different test models and data sets, a series of cube builds were performed on Windows NT and UNIX computers with various processor speeds (slower and faster). Keeping in mind that other hardware components do contribute to the total build time, the results clearly indicate that a faster CPU speed is better.

NT – Dual P200 *vs.* Dual Xeon 500



UNIX – Quad PA-RISC 8000 160MHz *vs.* Quad PA-RISC 9000 240MHz

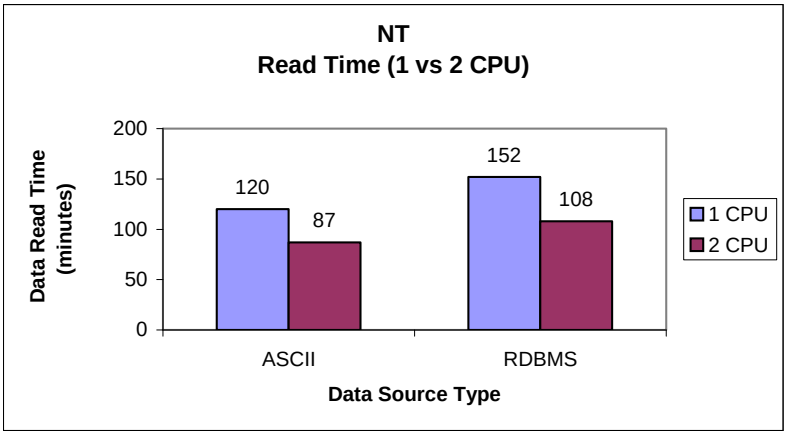


Note: Although the UNIX computers shown are Quad computers, Transformer is currently only capable of taking advantage of two processors per model, when multi-processing is enabled.

Examples of Read Time Reduction with 2nd CPU

The following test was done to illustrate the read time reduction that is obtained when a second CPU is available on the Transformer build computer. The '*Category and Row Dominant*' model (ASCII and RDBMS versions) was used to demonstrate the difference in build time on NT.

Note: The multi-processing feature available in Transformer must be enabled on each data source to take advantage of the second CPU.



Memory Considerations

Memory is probably the most important choice made during hardware selection, followed closely by disk configuration. The amount of memory selected can be

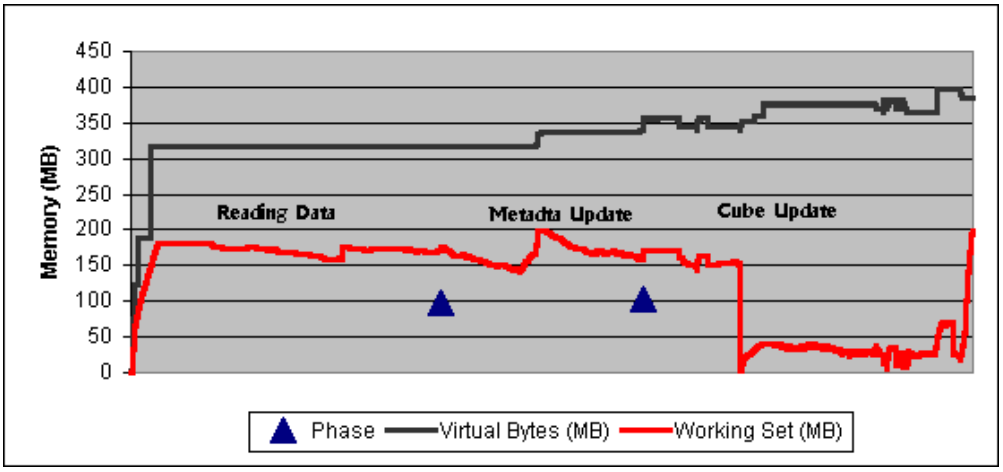
dictated by the number of categories in the Transformer model and the resulting size of the largest PowerCube (assuming that the server is dedicated to Transformer).

Optimally there should be enough memory on the build computer to handle all running application requests for memory and allow the operating system disk cache to grow as required during the PowerCube build. Excessive paging will take place in situations where there is not enough physical memory available for Transformer, which will result in a significant increase during PowerCube build time.

How Transformer uses Memory

As stated above, Transformer's memory consumption is directly related to the amount of categories in the model and the associated Transformer memory settings as selected by the Administrator.

The following is a chart that tracks Transformer's use of memory while processing the *'Category and Row Dominant'* test model:



The top line in the graph represents total 'Virtual bytes' used by Transformer while the lower one represents the 'Working Set'.

The 'Virtual Bytes' used by an application is the total amount of addressable memory the application has requested while the 'Working Set' represents the amount of physical memory that is actually being used by the application. The amount of memory represented by 'Working Set' comes out of the available physical memory on the computer.

Memory use climbs rapidly when categories are being generated during the Data Read phase as the data source is processed. The more categories, the more memory required. Memory use per category is not completely predictable because each model is different but observations of memory use for various models have shown that most fall in a range of 500 to 1,500 bytes per category (Working Set). Systems will have to resort to paging (swap file use) to continue processing when the amount of physical memory is limited and the 'Working Set' cannot grow to the amount required for the model. When this occurs the performance hit on PowerCube build is significant. For more information, please refer to the limited memory test chart in the following section.

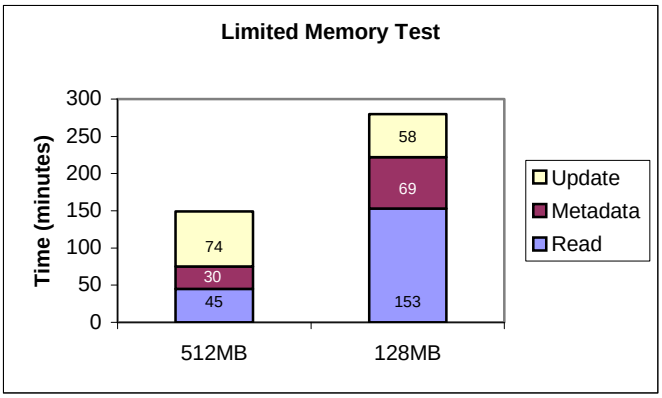
Memory use continues to be high through the Metadata Update stage but drops off significantly when data is being written to the PowerCube. At this stage, available

memory will be freed up and can be used by the operating system disk cache as required when the actual PowerCube is being built.

Limited Memory Testing

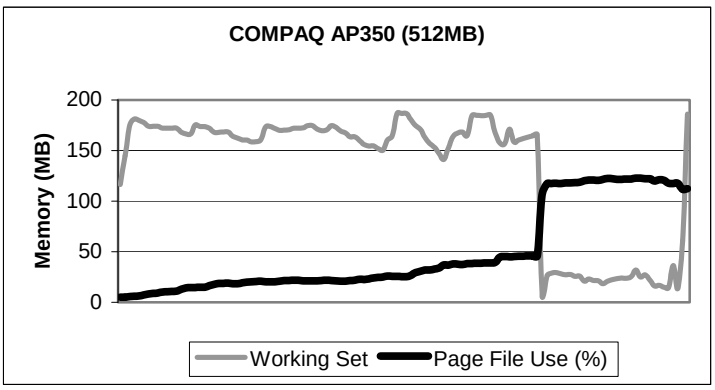
Using the *'Category and Row Dominant'* test model, a series of tests were run on the same NT computer (COMPAQ AP350) with different amounts of available system RAM to see what effect this would have on build time. First, the model was run with all available system memory (512MB) and the results recorded. The second test involved setting the amount of system RAM well *below* the working set recorded for the full memory test (128MB).

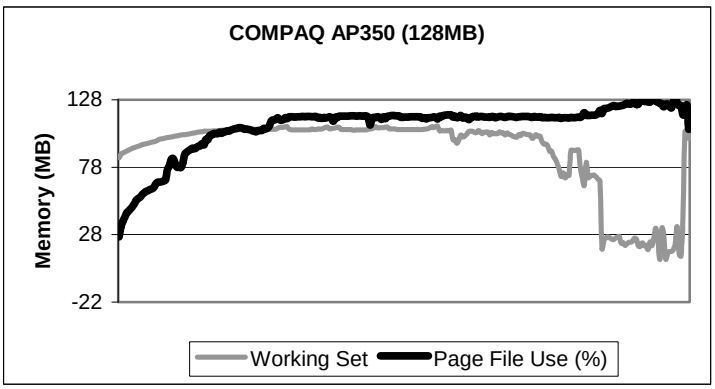
The following chart displays the timing results:



This particular test model has a 'Working Set' requirement of approximately 200MB. The chart shows that cube build time degrades considerably if the available physical memory on the computer is below Transformer's 'Working Set' requirement.

Another way to look at this is by looking at Page File activity during the two test runs. The first chart looks at 'Working Set' memory compared to the percentage of Page File use on the system for the test run with 512MB of available memory.





Note the difference of the Page File graph lines on the two charts. When you compare the two charts it is immediately evident that the Working Set is much smaller for the test run with only 128MB of RAM available. The smaller Working Set causes a significant increase in Page File use which has a negative effect on the time it takes to build the PowerCube.

Hard Drive Considerations

This section provides some information to optimize your environment in relation to Transformer.

RAID

When larger PowerCubes are being built, disk space requirements can be quite high. The type of drives and amount of disk space available will have a very big impact on the PowerCube build process. The ideal configuration would consist of a drive subsystem that has multiple disk controllers with the fastest possible disk drives configured for RAID 0 or RAID 1:

RAID Level	Description
0 (striping)	Optimized performance at the expense of data redundancy. Data is distributed among disks for performance, with no provision for redundancy. As a result, a disk crash can cause data to be lost across several disks.
1 (mirroring)	Emphasizes data redundancy at the expense of performance. Mirroring maintains multiple copies of data to ensure that, in the event of a disk failure, no data is lost.

RAID level 0 provides the fastest performance. In the event of a disk failure during a PowerCube build, the cube can be rebuilt from the original source data.

Drive Configuration

Transformer is an I/O bound application. The type, speed and configuration of the drive subsystem can cause a significant increase in the time it takes to build a PowerCube.

Choosing a drive configuration for Transformer is very similar to the way it is done for relational database servers. Ideally, the drive subsystem should have more than one physical disk (three or more is optimum). The typical database installation sees

applications and operating system on one disk, data on another and indexes on the third. With Transformer the breakdown would be as follows:

- 1st Controller: Operating System and applications
- 2nd Controller: Transformer Data Work directory
- 3rd Controller: Sort directory and PowerCube directory

Lets assume that the server consists of the following configuration regarding controllers:

- 1st Controller is drive C
- 2nd Controller is drive D
- 3rd Controller is drive E

According to the Transformer specifications on drive configurations, the following would apply:

- Drive C would contain the operating system and Transformer application
- Drive D would contain the location for the DataWorkDirectory
- Drive E would contain the locations for the ModelWorkDirectory and the CubeSaveDirectory.

The log file below illustrates the above settings:

```
PowerPlay Transformer Wed Sep 19 09:39:17 2001
LogFileDirectory=c:\transformer\logs
ModelSaveDirectory=c:\transformer\models\
DataSourceDirectory=c:\transformer\data\
CubeSaveDirectory=e:\transformer\cubes\
DataWorkDirectory=d:\temp\
ModelWorkDirectory=e:\temp\
```

How Transformer uses Disk Space

During a cube build Transformer uses disk space in the following fashion:

- Data Read phase: During this phase Transformer is reading the source data and creating a temporary work file based on the structure of the Transformer model.
- Metadata Update phase: After the source data is read, the temporary work file is processed to determine the status of categories in the cube. A copy of the temporary work file is created and gets processed. After processing is complete, the original work file is deleted and all valid categories are put into the PowerCube.
- Data Update phase: After the categories are added to the PowerCube, the data in the temporary work file is inserted into the cube. If the PowerCube is partitioned, the temporary work file is sorted and then inserted into the PowerCube. Depending on the PowerCube settings a number of passes through the temporary work file may be required.

How Much Disk Space?

It is possible to calculate the amount of disk space that Transformer will require for the temporary work files used while building the PowerCube. The one thing that *cannot* be predicted in advance is the final size of the PowerCube. This is due to the

amount of variables that contribute to the PowerCube size which are unique to each environment, data set and model configuration.

The amount of space used in temporary files can be calculated as long as the Transformer model being used has been fully defined and the number of input records is known.

The following spreadsheet formula can be used to estimate temporary work file disk space requirements:

Number of Dimensions	0	Total
Number of Dimension Views	0	Attached to PowerCube
Number of measures	0	Regular measures only
Number of input records	0	Sum of all datasources with measure values
WorkFileMaxSize	0	TRNSFRMR.INI setting
Total	**0**	**Size of Workfile in MB**

This spreadsheet assumes the following:

- Auto-partitioning has been used
- Calculated measures are not counted
- Only count dimension views that are actually attached to the PowerCube
- Can be used to formulate single or PowerCube groups

The spreadsheet formula will provide a good estimate of the disk space required for temporary work files but does not account for the PowerCube and model checkpoint files. While there is no reliable method to accurately predict PowerCube size, a good rule of thumb would be to add 20% of the estimated disk space required for temporary files. The size of the Transformer checkpoint file will be roughly equivalent to the 'Working Set' for the model. For more information, please refer to section 6.3.

To calculate the size of a model work file, double click on the attached spreadsheet above. To determine the WorkFileMaxSize to enter in the spreadsheet, divide the existing number (found in the trnsfrmr.ini file) by 1024 for KB and then 1024 for MB. For example, if the default WorkFileMaxSize setting is used it would be calculated as follows:

(2000000000/1024)/1024 = 1907

Example of Estimated Space Calculations vs. Actual Cube Build

Using the spreadsheet formula the estimated disk space required for the *'Category and Row Dominant'* test model worked out as follows:

Number of Dimensions	5	Total
Number of Dimension Views	0	Attached to PowerCube
Number of measures	3	Regular measures only
Number of input records	49000000	Sum of all datasources with measure values
WorkFileMaxSize	1907	TRNSFRMR.INI setting
Total	**7047**	**Size of Workfile in MB**

The above spread shows a 7GB work file is created during the PowerCube build. A test system was then set up with the Transformer Data Temporary file and Sort directory all pointing to the same directory location. All other Transformer directory

locations were pointed to another directory (on another disk drive) and the Windows NT performance monitor was used during the cube build to track the amount of disk space available.

Other Applications on the Build Computer

Since Transformer can be considered a memory- and I/O-bound application it is not desirable to have other applications running on the PowerCube build computer that place a demand on the system in these areas. We recommend that Transformer be located on a server dedicated solely to PowerCube builds, or that no other applications are active during the cube builds.

Setting up the Transformer Environment

NT

This section lists the settings specific to Transformer on Windows NT that should be considered for optimum performance.

- WriteCacheSize: The value for the write cache can affect PowerCube build time in a positive or negative way depending on how much memory is available. The best performance is achieved when enough physical memory is available so that the disk cache can grow to be as large as the final size of the PowerCube.

  You can change this setting in the Configuration Manager under Services - PowerPlay Data Services - Cache. The default value is set to 8192 (or 8MB). To change this, increase the number by increments of 1024. Increasing the write cache to 32768 (32MB) or 65536 (64MB) on a large system can provide performance improvements. However, increasing it to a very large number (i.e. 102400 or hundreds of megabytes) can degrade performance.

- SortMemory: This variable sets the amount of physical memory that is available when the data is sorted. Transformer sorts data for consolidation and auto-partitioning.

  The number you specify represents the number of 2K blocks used when sorting data. For example, setting a value of 5120 provides 5120 x 2K = 10MB of memory. The default value is set to 512. You can change the default in the Configuration Manager under Services - UDA - General. A good place to start is by changing the default value to equal 5120.

- TEMPFILEDIRS: Transformer uses this setting for the temporary sort file. This file is created whenever Transformer has to perform a sort operation.

  You can change the location in the Configuration Manager under Services - UDA - General. You can specify multiple directories separated by semicolons.

- MaxTransactionNum: Transformer inserts checkpoints at various stages when generating PowerCubes. The Maximum Transactions Per Commit setting limits the number of records held in a temporary status before inserting a checkpoint. The default setting is MaxTransactionNum=500000. The value specified is the maximum number of records that Transformer is to process before committing the changes to a PowerCube. The default can be changed in the Transformer Preferences dialog box under the General tab.

If errors occur during a cube build (i.e. TR0112 There isn't enough memory available) lower the MaxTransactionNum so that it commits more frequently and frees up drive space.

This setting can be increased to a higher number (such as 800000) to improve the cube build time but the results will vary dependant on the environment.

Note: The ReadCacheSize setting is not relevant to Transformer. This setting is specific to PowerPlay Enterprise Server and PowerPlay Client only.

UNIX

This section lists the settings specific to Transformer on UNIX that should be considered for optimum performance.

- PPDS_WRITE_MEMORY: The value for the write cache can affect PowerCube build time in a positive or negative way depending on how much memory is available. The best performance is achieved when enough physical memory is available so that the disk cache can grow to be as large as the final size of the PowerCube.

  You can change this setting using the PPDS_WRITE_MEMORY environment variable. The default value is set to 32768 (or 32MB). To change this, increase the number by increments of 1024. Increasing the write cache to 65536 (64MB) or 98304 (96MB) on a large system can provide performance improvements. However, increasing it to a very large number (hundreds of megabytes) can degrade performance.

- SORTMEMORY: This variable sets the amount of physical memory that is available when the data is sorted. Transformer sorts data for consolidation and auto-partitioning.

  The number you specify represents the number of 2K blocks used when sorting data. For example, setting a value of 5120 provides 5120 x 2K = 10MB of memory. The default value is set to 512. You can change this setting in the Configuration Manager under Services - UDA - General.

- TEMPFILEDIRS: Transformer uses this setting for the temporary sort file. This file is created whenever Transformer has to perform a sort operation.

  You can change the location specified for this setting in the Configuration Manager under Services - UDA - General. You can specify multiple directories separated by semicolons.

- MaxTransactionNum: Transformer inserts checkpoints at various stages when generating PowerCubes. The Maximum Transactions Per Commit setting limits the number of records held in a temporary status before inserting a checkpoint. The default setting is MaxTransactionNum=500000. The value specified is the maximum number of records that Transformer is to process before committing the changes to a PowerCube. The default can be changed in the trnsfrmr.rc file or set in a Preference file.

  If errors occur during a cube build (i.e. TR0112 There isn't enough memory available) lower the MaxTransactionNum so that it commits more frequently and frees up drive space.

  This setting can be increased to a higher number (such as 800000) to improve the cube build time but the results will vary dependant on the environment.

- Ulimit: Most UNIX systems are configured to provide the best possible sharing of system resources between competing processes. This is not an optimal setting for Transformer, which requires as much physical memory as possible.

  For example, a HPUX server might have 2GB of physical memory, but be configured so that no single process can ever obtain more than 67MB. In such cases, Transformer will never obtain the memory required to perform large PowerCube builds in an efficient way. To ensure that Transformer can obtain the memory it requires, ensure that your UNIX server is configured to grant unlimited resources to the RSSERVER process by setting the ulimit to unlimited. To check the ulimit settings type ulimit -a on the UNIX command line. The following will be displayed:

```
    time(seconds)        unlimited
    file(blocks)         unlimited
    data(kbytes)         65536
    stack(kbytes)        8192
  memory(kbytes)         unlimited
    coredump(blocks)     4194303
    nofiles(descriptors) 1024
```

  For best results the memory(kbytes) option should be set to unlimited.

  For other UNIX platforms, contact the system administrator or the operating system documentation to determine how to tune your kernel settings to allow Transformer to have enough physical memory.

  Note: The ReadCacheSize setting is not relevant to Transformer. This setting is specific to PowerPlay Enterprise Server and PowerPlay client only.

Running Multiple Instances of Transformer

If the server is a multi CPU system, multiple instances of Transformer can be run in parallel. This is especially useful when a large number of cubes must be built in parallel to meet a PowerCube build production window.

When running multiple Transformer instances, the following is recommended:

- Each Transformer *process* should have its own dedicated CPU. If Multi-Processing is enabled, then each instance of Transformer should have 2 dedicated CPUs.
- Each Transformer instance will use system resources independent of all other instances. Ensure that you have sufficient memory, disk space, and I/O bandwidth to support all instances.
- Each Transformer instance will require its own set of configuration files. It is recommended that the DataWorkDirectory and ModelWorkDirectory locations are *not* shared between Transformer instances. For more information on how to set up the configuration files, please refer to section 6.9.

Tips

- Using the UNIX nohup command will allow you to continue executing the command even though you have logged out of the session. Example:

```
nohup rsserver -mmodel.mdl
```

- Adding an ampersand (&) to the end of the UNIX command line will allow you to start the first process in the background giving you back control of the prompt which will allow you to initiate the second RSSERVER command.

```
rsserver -mmodel.mdl &
```

Preference Files

When Transformer begins a PowerCube build, the model is populated with categories, cubes are generated and a log file is created. How and where these actions are performed is determined by a number of preferences and environment settings that you can specify in preference files.

Several preference file settings are available for use but the most commonly used ones are listed below:

- **ModelWorkDirectory=<path>**
  Specifies where Transformer creates temporary files while you work on your model. The temporary file can be used to recover a suspended model at strategic checkpoints should a severe error occur during cube creation. This file has the extension QYI. The default path is the value of the ModelSaveDirectory setting.

- **DataWorkDirectory=<path1;path2;...>**
  Specifies where Transformer creates temporary work files while generating cubes. Being able to use multiple drives eliminates size limitations set by the operating system. As Transformer creates cubes it writes temporary files to the specified drives or directories. The files are then concatenated into one logical file, regardless of which drive they are in. The location of these files is determined by the list of paths that you specify. The default path is the value of the CubeSaveDirectory setting.

- **DataSourceDirectory=<path>**
  For data source files other than IQD files and Architect models, this setting specifies where Transformer searches for the files. The default path is the current working directory.

- **CubeSaveDirectory=<path>**
  Specifies where Transformer saves cubes. The default path is ModelSaveDirectory.

- **ModelSaveDirectory=<path>**
  Specifies where Transformer saves models. The default path is the current working directory.

Here is an example of these settings in a Transformer log file:

```
PowerPlay Transformer Wed Sep 19 09:39:17 2001
```

```
LogFileDirectory=c:\transformer\logs
ModelSaveDirectory=c:\transformer\models\
DataSourceDirectory=c:\transformer\data\
CubeSaveDirectory=e:\transformer\cubes\
DataWorkDirectory=d:\temp\
ModelWorkDirectory=e:\temp\
```

The examples below display how to specify the use of a preference file on the command line:

Windows:

trnsfrmr -n -fc:\preferences.prf model.mdl

UNIX:

rsserver -F preferences.rc –mmodel.mdl

Tips

- Specifying the use of a preference file on the command line will override and take precedence over all other settings. For example, if you have environment settings defined in the rsserver.sh file, using a preference file on the command line will override these settings.
- The environment variables TMPDIR, TEMP, and TMP can also determine where Transformer creates temporary files. Transformer uses the first environment variable that is defined. These environment variables are system environment variables defined by the operating system.

Database Gateway Settings

A number of gateway INI files are included with a Transformer install that include database specific settings that can help reduce the read phase during a cube build. All files are named COGDM*.INI with the asterisk representing a specific database version of this file. For example, the Oracle specific INI file is named COGDMOR.INI and is located in the <install>\cer2 directory.

This file contains the following settings:

- Fetch Number of Rows: This setting is used to determine how many rows to fetch per fetch operation. Increasing this number can provide better performance on some systems. Note that this number is currently limited to 32767. Also note that numbers larger than 100 may actually degrade performance on some systems:

  Fetch Number of Rows=100

- Fetch Buffer Size: This setting is used to determine the size of buffer to use when fetching. Larger values can provide better performance on some systems. By default, the buffer size used is 2048 bytes, to change this default, edit the following entry and set it accordingly:

  Fetch Buffer Size=2048

Note: If Fetch Buffer Size and Fetch Number of Rows are both set, Fetch Number of Rows will take precedence.

Resolving Issues

This section concentrates on some common issues that may arise when building large PowerCubes, along with some suggestions to try and resolve them.

Three Phases of PowerCube Builds

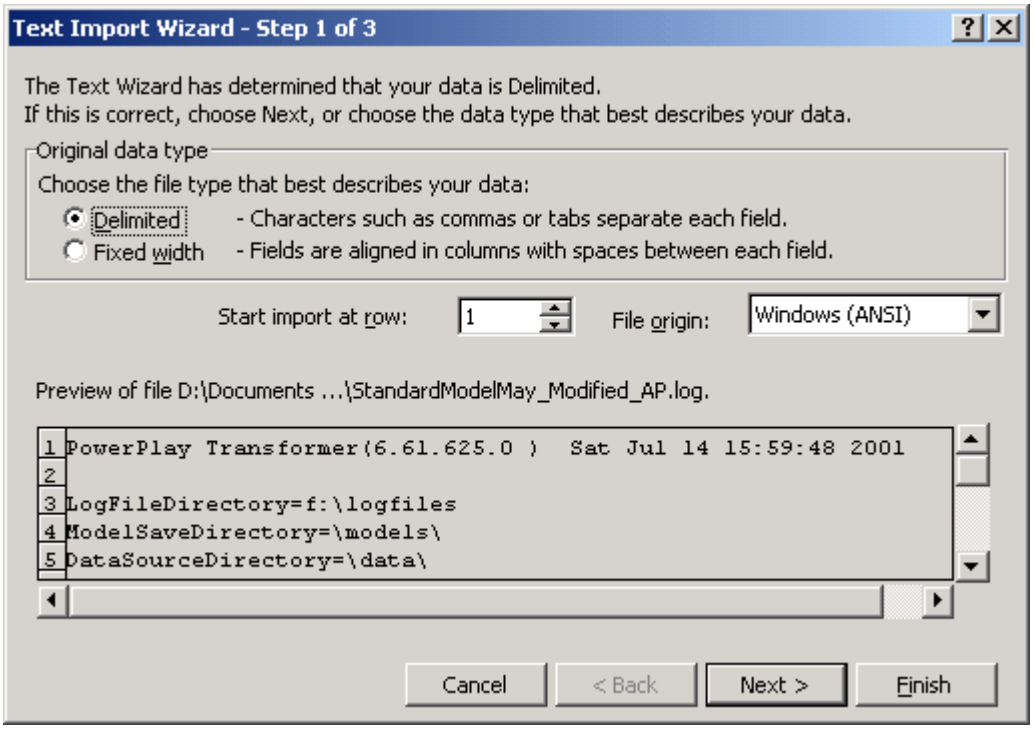
It is important that the user have a good understanding of the three distinct phases Transformer goes through to build a PowerCube. It is also important to determine how long each of these phases takes for a particular PowerCube build if the issue is related to timing. The three phases of a cube build are:

- Data Read: During this phase the input records are read from the selected data source into temporary work files. Common issues during this phase include database connectivity and insufficient disk space.
- Metadata Update: During this phase the contents of the temporary work files are compared to the categories in the Transformer model to determine which categories will be put in the PowerCube. When the list of eligible categories is complete the categories are inserted into the PowerCube. Common issues during this phase include lack of memory and insufficient disk space.
- Data Update: During this phase the actual data values in the temporary work files are inserted into the PowerCube. Each record inserted into the cube is a 'data point' that consists of a category reference from each dimension in the model along with the measure values for the intersection of those categories. A common issue during this phase is low system memory.

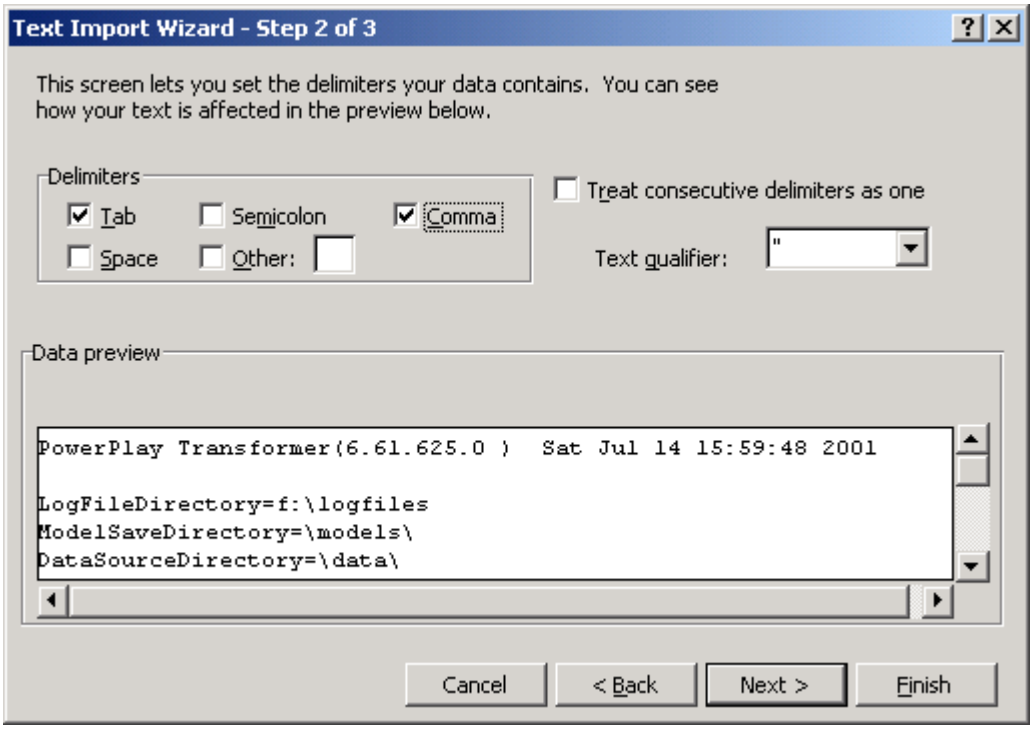
Using the Transformer Log File for Phase Timing

A Transformer log file is generated every time a model is processed. Using a spreadsheet program this log file can be used to quickly understand the length of each of the three phases of a cube build. Here is an example of how to do this:

1. Launch a spreadsheet program (Excel was used for this example) and select File Open for files of type 'All Files (*.*)'.
2. The dialog for Step 1 of the Text Import Wizard appears:

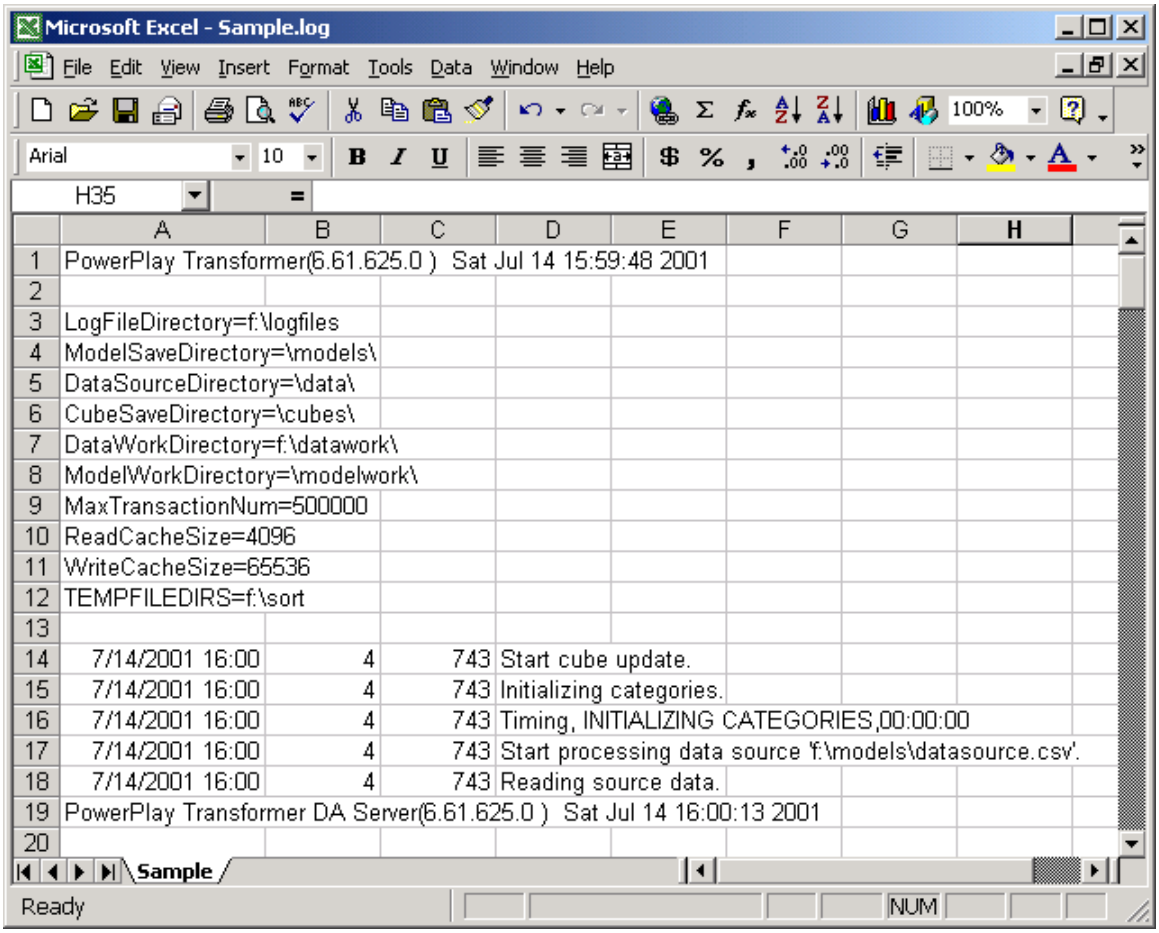


3. Keep the type as ‘Delimited’ and select ‘Next’. The dialog for Step 2 of the Text Import Wizard appears:

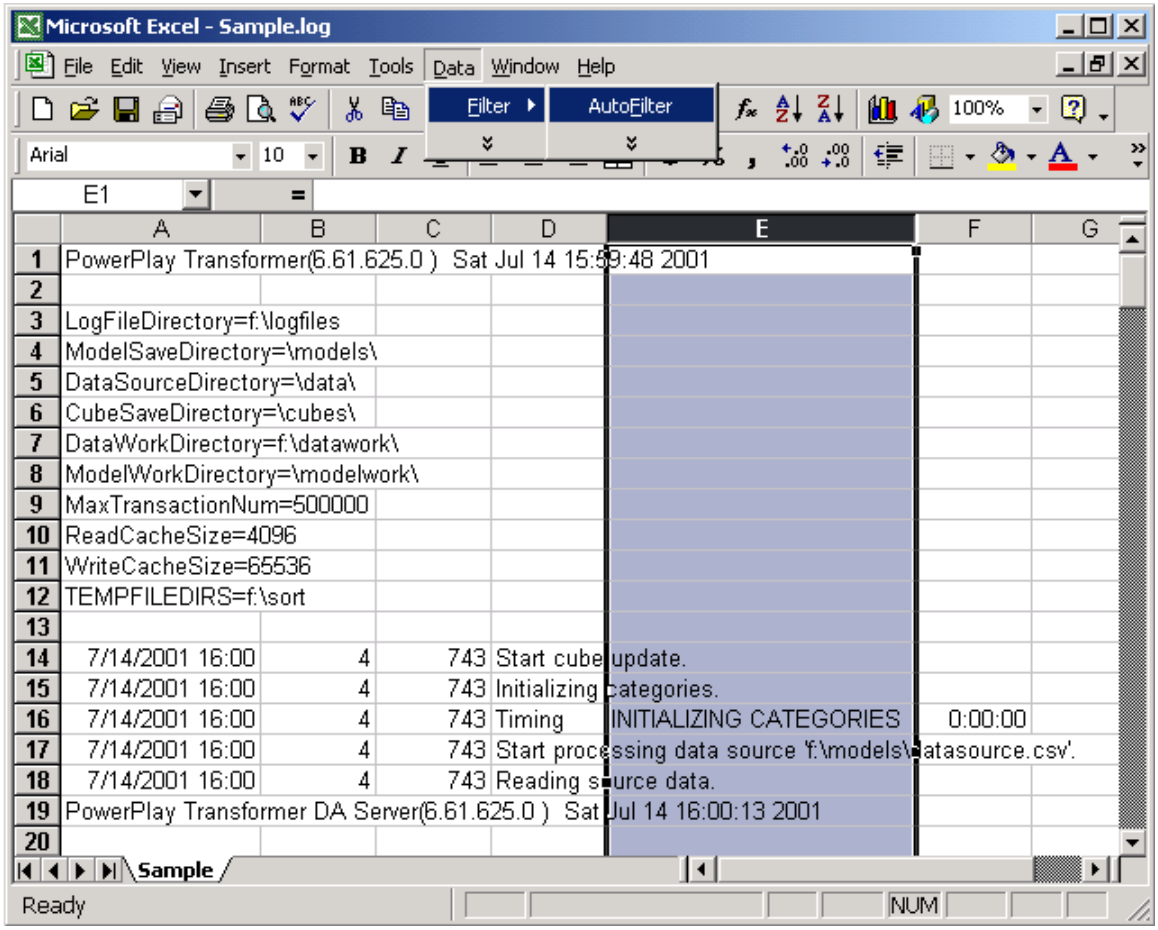


4. Make sure that ‘Tab’ and ‘Comma’ are selected, then click ‘Finish’.

5. The log file is loaded into Excel and appears as follows:



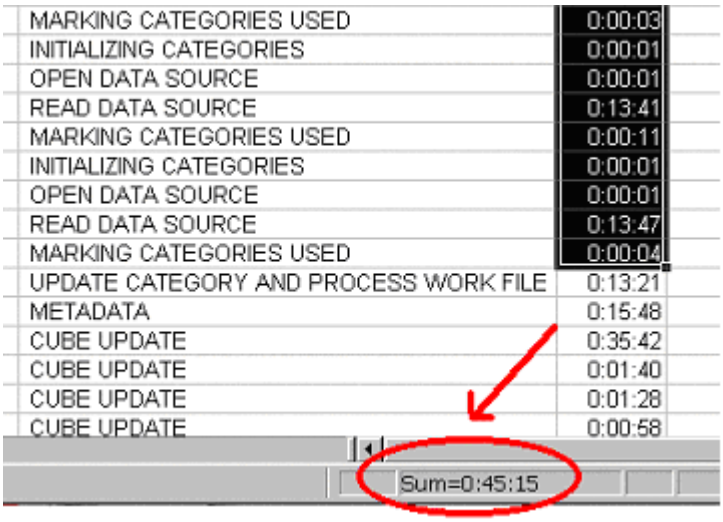
6. Select the entire ‘E’ column (by selecting the header) and then choose the ‘Data’ menu item followed by the ‘Filter’ item and finally select the ‘AutoFilter’ option:



7. From the drop down list that appears in the ‘E’ column select ‘(NonBlanks)’ or a specific phase such as Read Data Source or Metadata.

8. The spreadsheet now shows only the lines that contain timing information.

9. Once the spreadsheet is in this format, select a range of cells in the 'F' column and look at the bottom of the Excel window to see the sum of the timing values:



10. The following displays the keywords that relate to each of the phases of a PowerCube build:

    Data Read
    - INITIALIZING CATEGORIES
    - OPEN DATA SOURCE
    - READ DATA SOURCE
    - MARKING CATEGORIES USED

    Metadata Update
    - SORTING
    - UPDATE CATEGORY AND PROCESS WORK FILE
    - METADATA

    Data Update
    - CUBE UPDATE
    - CUBE COMMIT

11. If the default PowerCube optimization is used then the phases will appear in distinct sequential blocks (each phase completes before proceeding to the next). With older cube optimizations it is possible to see phases repeated (i.e. Read, Metadata, Update, Read, Metadata …)

12. Sometimes timing shown for the TOTAL TIME (CREATE CUBE) keyword will be different than the timing for the individual phases. If this happens simply adjust the time difference to the Cube Update phase.

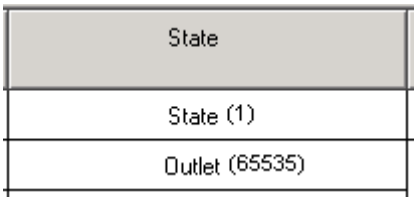
Looking at the Transformer log files in this manner can help determine a how much time is being spent on each phase. You may notice over time that PowerCube builds are taking longer to complete although the dataset is relatively the same. Comparing log files can help you determine where the increase in build time is occurring.

Supported Limits

This section discusses the current limitations that exist in Transformer Series 7.

Parent:Child Ratio

Currently the parent:child ratio limit is 1:65535. This limit exists as part of the architectural design and is specific to the number of categories that exist between levels. The example below demonstrates this limit:



Warning! It is not recommended that the hierarchical ratio extend to the full limit of 1:65535 due to possible performance related issues.

ASCII File Size

Prior to Series 7 Version 2, a limit on the physical file size of ASCII files existed. The maximum size that Transformer could process was 2GB. It was possible to bypass this limit in previous versions by using several ASCII files specified as individual data sources within the Transformer model.

Number of Categories

With Series 7, the estimated maximum number of categories is approximately 2 Million. This number isn't a fixed limit as many factors contribute to the maximum number of categories that can be obtained for each model. These factors, also known as modeling choices are many.

The Transformer model stores metadata about the data source objects, otherwise referred to as categories. These categories are derived from the structural data sources specified in the model. The metadata contained in the model, impacts the storage (file) size of the Transformer model. A list of the modeling choices that impact the storage size include:

- Labels
- Descriptions
- Short Names
- Category Codes
- Dimensions Views
- User Class Views

For example, if large descriptions are used in the model, these descriptions can require a lot of storage space. If descriptions are not required, they can be eliminated from the model.

In combining all of these modeling choices, a limitation may be reached:

- File size - there is an operating system file size limitation of 2GB on the Transformer model
- Virtual memory – the Transformer model is held entirely in allocated memory meaning there is a limit imposed by the amount of address space available
- Physical memory - the Transformer model is held entirely in allocated memory and a severe performance penalty will occur if the Transformer model is larger than the physical memory available

Case Studies

The following case studies are included to provide the reader with some insight into the dramatic differences in PowerCube build times when various factors are taken into consideration. These case studies consist of actual client test cases.

Note: All case studies were performed in an isolated lab where external influences were not a concern. No other applications were active during the cube builds in the BEFORE or AFTER tests and case studies.

Case Study #1

Based on a number of factors including number of transactional records, it was apparent that this PowerCube build was taking a long time to complete, which warranted an investigation.

Description of Model:

Model Attribute	Description
Number of Categories	546,391
Number of Dimensions	9 (measures not counted as a dimension)
Number of Measures	10 - four calculated: "Category A" * 100 / "Category B" "Category C" * 100 / "Category B" "Category D" * 100 / "Category B" "Category E" * 100 / "Category B"
Source Data Format	ASCII (',' delimited)
Number of source files	23 (13 are structural, 10 are transactional) Multi-Processing was enabled for all data sources.
Number of Transaction input records	3 million (2,993,031)
Size (in MB) of all source files	1.28GB

Original Transformer Log File (BEFORE):

Phase	Time
READ DATA SOURCE	28 minutes
METADATA	2 hours, 10 minutes
CUBE COMMIT	4 hours, 49 minutes
TOTAL TIME (CREATE CUBE)	7 hours, 41 minutes

Diagnosis:

During an analysis of the log file the following warning was discovered:

> Warning: (TR2757) This model contains one or more cubes that use a dimension view in which the primary drilldown is cloaked. Auto-partitioning is not possible when a primary drilldown is cloaked.

As mentioned previously in this document, disabling auto-partitioning can have a significant impact on build time. Please refer to section 5.5 for more details. After changing the primary drill category we resolved the above warning.

Updated Transformer Log File (AFTER):

Phase	Time
READ DATA SOURCE	28 minutes
METADATA	4 minutes
CUBE COMMIT	4 minutes
TOTAL TIME (CREATE CUBE)	41 minutes

Conclusion:

By making one small change in the model, the build time decreased dramatically from 7 hours and 41 minutes to 41 minutes.

Case Study #2

As the PowerCube became increasingly longer to build and larger in size, it was necessary to optimize performance by changing hardware so the cube could be built within the production window.

Description of Model:

Model Attribute	Description
Number of Categories	497,640
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 - two calculated: ("Category A" - "Category B") / 100 "Category A" - "Category B"
Source Data Format	ASCII ('~' delimited)
Number of source files	9 (6 are structural, 3 are transactional) Multi-Processing was enabled for the 4 largest data sources.
Number of Transaction input records	250 million (248,042,742)
Size (in MB) of all source files	2.28GB

Original Transformer Log File (BEFORE):

Phase	Time
READ DATA SOURCE	11 hours, 50 minutes
METADATA	21 minutes
CUBE UPDATE	20 hours
CUBE COMMIT	30 minutes
TOTAL TIME (CREATE CUBE)	35 hours, 16 minutes

Diagnosis:

Purchased a new server dedicated to Transformer cube builds. This case study dramatically proves how hardware can affect the total cube build time.

Original Server Specs:

Digital Prioris Model ZX-109
Dual 200MHz Pentium Pro Processor
512 MB RAM
160GB HD

New Server Specs:

CCPQ ProLiant ML570 RM 700 2 GB
Intel Pentium III 700 MHz processor x 1 Quad capability
Cache Memory: 1-MB level 2 writeback cache per processor
Memory: PC100-MHz Registered ECC SDRAM DIMM SDRAM DIMM standard: 512 MB (4 x 128MB
CCPQ 2GB PC100 Registered ECC SDRAM (4 x 512MB)
Optical Drive: High Speed IDE CD-ROM Drive (Low Profile)
CCPQ PIII Xeon 700MHz Processor Option Kit
CCPQ Smart Array Controller 5302/64
CCPQ 18.2 GB Wide Ultra 3 15,000rpm (1") HP Drives
CCPQ Secure Path Software V3.0 for Win NT (v3.1?)
CCPQ 64bit PCI to Fiber Channel Host Bus Adapter, NT

Updated Transformer Log File (AFTER):

Phase	Time
READ DATA SOURCE	3 hours, 5 minutes
METADATA	4 minutes
CUBE UPDATE	2 hours, 42 minutes
CUBE COMMIT	5 minutes
TOTAL TIME (CREATE CUBE)	6 hours, 27 minutes

Conclusion:

The hardware being utilized to build PowerCubes can have a dramatic effect as this example demonstrates.

Case Study #3

In order to build a large number of cubes within a specified time frame, it became necessary to have multiple instances of Transformer running building PowerCubes.

Description of Models:

Model A:

Model Attribute	Description
Number of Categories	492,152
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 - two calculated: ("Category A" - "Category B") / 100 "Category A" - "Category B"
Source Data Format	ASCII ('~' delimited)
Number of source files	9 (6 are structural, 3 are transactional) Multi-Processing was enabled for the 4 largest data sources.
Number of Transaction input records	50 million (49,540,177)
Size (in MB) of all source files	2.28GB

Model B:

Model Attribute	Description
Number of Categories	146,238
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	15
Source Data Format	ASCII ('~' delimited)
Number of source files	6 (5 are structural, 1 is transactional) Multi-Processing was enabled for the 5 largest data sources.
Number of Transaction input records	1 million (970,000)
Size (in MB) of all source files	224MB

Model C:

Model Attribute	Description
Number of Categories	546,391
Number of Dimensions	9 (measures not counted as a dimension)
Number of Measures	10 (four calculated)
Source Data Format	ASCII (',' delimited)
Number of source files	23 (13 are structural, 10 are transactional) Multi-Processing was enabled for all data sources.
Number of Transaction input records	9 million (9,046,382)
Size (in MB) of all source files	1.28GB

Model D:

Model Attribute	Description
Number of Categories	33,145
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	4 (one calculated)
Source Data Format	ASCII (',' delimited)
Number of source files	14 (11 are structural, 4 are transactional) Multi-Processing was enabled for 3 data sources.
Number of Transaction input records	10 million (9,907,682)
Size (in MB) of all source files	569MB

Individual Build Times (BEFORE):

Model	PowerCube Build Time
Model A	2 hours, 24 minutes
Model B	12 minutes
Model C	11 hours, 31 minutes
Model D	16 minutes
TOTAL TIME (CREATE CUBE)	14 hours, 25 minutes

Concurrent Build Times (AFTER):

Model	PowerCube Build Time
Model A	2 hours, 41 minutes
Model B	14 minutes
Model C	12 hours, 2 minutes
Model D	19 minutes
TOTAL TIME (CREATE CUBE)	12 hours, 2 minutes

Conclusion:

Having a server with 8 CPUs allows you the flexibility of running four PowerCube builds at the same time (with Multi-Processing enabled in each model). Building the PowerCubes concurrently saved 2 hours and 23 minutes off of the total build times in comparison to building the PowerCubes individually.

Case Study #4

As the PowerCube became increasingly longer to build and larger in size, it was necessary to optimize performance by changing hardware so the cube could be built within the production window.

Description of Model:

Model Attribute	Description
Number of Categories	492,152
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 - two calculated: ("Category A" - "Category B") / 100 "Category A" - "Category B"
Source Data Format	ASCII ('~' delimited)
Number of source files	9 (6 are structural, 3 are transactional) Multi-Processing was enabled for the 4 largest data sources.
Number of Transaction input records	50 million (49,540,177)
Size (in MB) of all source files	2.28GB

Original Transformer Log File (BEFORE):

Phase	Time
READ DATA SOURCE	1 hour, 53 minutes
METADATA	29 minutes
CUBE UPDATE	2 hours, 39 minutes
CUBE COMMIT	7 minutes
TOTAL TIME (CREATE CUBE)	5 hours, 27 minutes

Diagnosis:

Purchased a new server dedicated to Transformer cube builds. This case study dramatically proves how hardware can affect the total cube build time.

Original Server Specs:

SUN Microsystems Enterprise 250 S/N 913H202F
Dual 300 MHZ/2MB CPU
1 GB RAM

New Server Specs:

SUN Microsystems Sunfire 4800
8 x 750 MHZ Ultra-SPARC III processors
9.6 GB/sec sustained bandwidth
16 GB RAM

Updated Transformer Log File (AFTER):

Phase	Time
READ DATA SOURCE	38 minutes
METADATA	5 minutes
CUBE UPDATE	27 minutes
CUBE COMMIT	3 minutes
TOTAL TIME (CREATE CUBE)	1 hour, 23 minutes

Conclusion:

The hardware being utilized to build PowerCubes can have a dramatic effect as this example demonstrates.

Case Study #5

This case study is meant as a way to show the exponential increase in various facets including build time, cube size, etc.

Server Specs:

CCPQ ProLiant ML570 RM 700 2 GB
Intel Pentium III 700 MHz processor x 1 Quad capability
Cache Memory: 1-MB level 2 writeback cache per processor
Memory: PC100-MHz Registered ECC SDRAM DIMM SDRAM DIMM standard: 512 MB (4 x 128MB
CCPQ 2GB PC100 Registered ECC SDRAM (4 x 512MB)
Optical Drive: High Speed IDE CD-ROM Drive (Low Profile)
CCPQ PIII Xeon 700MHz Processor Option Kit
CCPQ Smart Array Controller 5302/64
CCPQ 18.2 GB Wide Ultra 3 15,000rpm (1") HP Drives
CCPQ Secure Path Software V3.0 for Win NT (v3.1?)
CCPQ 64bit PCI to Fiber Channel Host Bus Adapter, NT

Description of Models:

Model A:

Model Attribute	Description
Number of Categories	499,874
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 - two calculated: ("Category A" - "Category B") / 100 "Category A" - "Category B"
Source Data Format	ASCII ('~' delimited)
Number of source files	32 - Multi-Processing was enabled for each data source
Number of Transaction input records	500 million (491,249,782)
Size (in MB) of all source files	22.9 GB
Cube Size	2.2 GB

Model A Transformer Log File:

Phase	Time
READ DATA SOURCE	5 hours, 53 minutes
METADATA	5 minutes
CUBE UPDATE	5 hours, 5 minutes
CUBE COMMIT	5 minutes
TOTAL TIME (CREATE CUBE)	12 hours, 15 minutes

Model B:

Model Attribute	Description
Number of Categories	501,707
Number of Dimensions	5 (measures not counted as a dimension)
Number of Measures	5 - two calculated: ("Category A" - "Category B") / 100 "Category A" - "Category B"
Source Data Format	ASCII ('~' delimited)
Number of source files	68 – Multi-Processing was enabled for each data source
Number of Transaction input records	1 billion (1,000,532,636)
Size (in MB) of all source files	45.5 GB
Cube Size	4.6 GB

Model B Transformer Log File:

Phase	Time
READ DATA SOURCE	12 hours, 18 minutes
METADATA	5 minutes
CUBE UPDATE	11 hours, 25 minutes
CUBE COMMIT	10 minutes
TOTAL TIME (CREATE CUBE)	26 hours, 11 minutes

Conclusion:

Comparing the results of these two builds demonstrates the increase in build time and cube size as the number of source records and categories increase.

Case Study #6

This case study represents an actual test as performed during beta testing for an existing IBM Cognos PowerPlay customer. We compared the build time of an Incrementally Updated PowerCube to a Time-Based Partitioned Cube.

Description of Model:

Model Attribute	Description
Number of Categories	73,000
Number of Dimensions	6 (measures not counted as a dimension)
Number of Measures	28 - eight calculated
Source Data Format	IQDs
Number of source files	9 - Multi-Processing was enabled for all data sources
Number of Transaction input records	2 million (Incremental Update) 1.1 million (Time-Based Partitioned Cube)

Incremental Update Log File (BEFORE):

Phase	Time
READ DATA SOURCE	23 minutes
METADATA	20 minutes
CUBE COMMIT	6 minutes
DATA UPDATE	11 hours
TOTAL TIME (CREATE CUBE)	12 hours, 5 minutes

Diagnosis:

When an incremental update is performed several Data Updates occur because auto-partitioning is no longer happening. This results in a slower cube build.

The Time-Based Partitioned Cube feature not only takes advantage of auto-partitioning but the cube builds are much faster as the Data Update phase is not used.

Time-Based Partitioned Cube Log File (AFTER):

Phase	Time
READ DATA SOURCE	7 minutes
METADATA	22 seconds
CUBE COMMIT	5 seconds
DATA UPDATE	0
TOTAL TIME (CREATE CUBE)	14 minutes

Conclusion:

By modifying the model to take advantage of Time Based Partitioned Cubes, the build time decreased dramatically from 12 hours to 14 minutes.

NOTE: Although the number of data source records differ between the Incremental Update and the Time-Based Partitioned Cube builds, we believe the results can still be meaningfully compared.